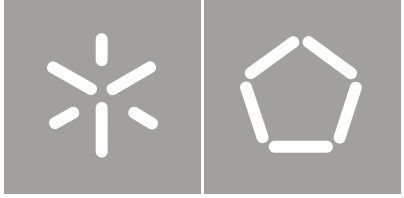
Universidade do Minho
Escola de Engenharia

Carlos Manuel Rodrigues Machado

Autonomic Ubiquitous Computing: A Home Environment Management System

Dezembro de 2009



Universidade do Minho
Escola de Engenharia

Carlos Manuel Rodrigues Machado

Autonomic Ubiquitous Computing: A Home Environment Management System

PhD Thesis in Industrial Electronics
Area of Knowledge in Industrial Computing

Work carried out under guidance of
José A. Mendes, PhD
João L. Monteiro, PhD

Dezembro de 2009

DECLARAÇÃO

Nome: Carlos Manuel Rodrigues Machado

Endereço Electrónico: carlos.machado@dei.uminho.pt **Telefone:** +351 967 281 949

Nº de Bilhete de Identidade: 10528984

Titulo da Tese de Doutoramento:

Autonomic Ubiquitous Computing: A Home Environment Management System

Orientadores:

José A. Mendes, PhD

João L. Monteiro, PhD

Ano de Conclusão: 2009

Ramo de conhecimento do Doutoramento:

Industrial Electronics, Area of Knowledge in Industrial Computing

Universidade do Minho, ___/___/_____

Assinatura: _____

To my family

(À minha família)

"No problem is too small or too trivial
if we can really do something about it."
Richard Feynman

Acknowledgements

The accomplishment of a task not only depends on the person who performs it, but also and always on a set of people who contribute and enable its development. Thus, I would sincerely like to acknowledge all the persons, institutions and companies for their valuable cooperation, having contributed to this work, in particular:

To the Portuguese Foundation for Science and Technology (FCT), for the financial support given with the scholarship number SFRH/BD/8290/2004.

To José A. Mendes, PhD, João L. Monteiro, PhD, and Julie A. McCann, PhD, for the thesis proposal and for providing me with the opportunity to develop this work.

To José A. Mendes, PhD, and Isabel Cunha, Msc, for reviewing the writing up of this thesis and associated publications.

To the companies, Ivv-Automação, Lda. and Ribeiro&Machado, Lda, for the hardware sponsorship that was used in the case-study house.

To Vasco Macedo and José Machado, for the help and practical know-how in the re-wiring of the electrical installation in the case-study house.

To Beatriz Antunes and Paula Anjo for being always available and helping in all the bureaucracy work that has been required by all institutions during throughout this project.

And finally to: Carlos, Joel, Angela, Isabel, João, Carla, Elisabete and especially to Jaime Gomes and Pedro Vieira for the encouragement, suggestions, availability, confidence, understanding and recommendations that have pushed me towards the conclusion of this work.

Abstract

The Ubiquitous Computing and Autonomic Computing reached a point of convergence in which pervasive technology in the environment meets the ability of people to interact with it, making use of all the possibilities made available by this technology. Ubiquitous computing envisions a habitat where the abundance of devices, services and applications allows the physical and virtual worlds to become seamlessly merged. The promise of ubiquitous computing environments is not feasible unless these systems can effectively "disappear". In order to achieve this goal, they need to become autonomic, by managing their own evolution and configuration with minimal user intervention. It is in this context that aspects like self-configuration and self-healing from the autonomic computing concept were adopted in this project.

The context awareness and the creation of applications which use that context are the core concern of Ubiquitous Computing Systems and represent the fundamentals for autonomic actions in this type of systems. Such research raises questions on context acquisition, distribution and manipulation, as well as on artificial intelligence algorithms that decide autonomic actions in the environment, having implications in the human interaction with Autonomic Ubiquitous Systems.

The research presented in this thesis concentrates on some of those issues. During this project it was developed an experimental setup for context acquisition, in an effortless way, of some activities of a small group of users. This experimental setup was installed in a real home where a young family, a couple and a small child, were actually living. This experimental setup was mainly responsible for the control of the light system of the house, by a network of several inter-connected devices scattered in the home with limited resources. This prototype installation allowed the validation of the system ability, to capture daily life behaviour patterns of the inhabitants.

The system architecture was designed based on the concept of a high level and a low level autonomic management system taking from nature the model of the human reflex arc. A reflexive behaviour is managed at a local level by the small devices, with limited resources, high level management is responsible for processing and analysis of the events broadcast by the group of small devices, and run in a centralized mode in a PC.

The concept of device information broadcast, to the communication medium, as events was used as an approach to: inter-connect future systems, monitor correct operation of the system devices, capture raw data for estimation of context; allow the visualization of system feedback in user interface devices.

Finally, an algorithm using artificial neural networks in combination with simple statistics was developed which allowed the house to learn the routines of its inhabitants, making it truly intelligent by embedding the knowledge about patterns of activities of the users in the devices scattered in the environment, increasing their comfort and, at same time, leading to more energy efficiency. The analysis of the data captured, during two complete years, shows that the reduction of power consumption could be as high as 50%, depending on the profile of the usage of the light.

Resumo

A Computação Ubíqua e a Computação Autónoma atingiram um ponto de convergência no qual a tecnologia dispersa nos ambientes, juntamente com a capacidade das pessoas interagirem, permite tirar partido do seu uso para novas potencialidades. A computação ubíqua vislumbra habitats repletos de dispositivos, serviços e aplicações que permitem a união perfeita do mundo real com o mundo virtual, mas de uma forma natural. A promessa da criação de tais ambientes de computação ubíqua não se tornará possível sem que a complexidade destes sistemas "desapareçam" efectivamente da percepção dos utilizadores. Para que isso seja possível, estes necessitam de ser autónomos, gerindo a sua própria evolução e configuração com o mínimo de intervenção do utilizador. É neste contexto que a noção de Sistemas Ubíquos Autónomos envolvendo as facetas de auto-configuração e auto-reparação derivadas do conceito da computação autónoma, será usada nesta tese.

A percepção do contexto e a criação de aplicações que o usam são as principais preocupações na investigação dos Sistemas de Computação Ubíqua, constituindo também a base para as acções autónomas neste tipo de sistemas. Essa investigação levanta questões sobre a forma como o contexto é capturado, distribuído e manipulado. Por outro lado, provoca impacto nos algoritmos de inteligência artificial que efectuam as decisões de acções autónomas no ambiente, afectando consequentemente a interacção humana com os Sistemas Ubíquos Autónomos.

A investigação apresentada nesta dissertação concentra-se efectivamente em alguns destes aspectos. Durante a tese foi desenvolvido um sistema experimental com o objectivo de capturar o contexto, de uma forma perceptível, das actividades de um pequeno grupo de utilizadores. Este sistema experimental foi instalado numa casa real, onde vive uma jovem família constituída por uma casal e uma pequena criança. O sistema experimental era responsável por controlar toda a iluminação eléctrica da casa, através de um conjunto de dispositivos, com recursos limitados, conectados em rede e espalhados pela casa. A instalação permitiu validar a capacidade do sistema de capturar os padrões de comportamento quotidiano dos habitantes da casa.

A arquitectura do sistema foi projectada baseando-se no conceito de alto-nível e baixo-nivel dos sistemas de gestão autónoma, inspirando-se no modelo dos processos que ocorrem num acto

reflexo no corpo humano. As acções de reflexo ou acções básicas são geridas pelo baixo-nivel nos pequenos dispositivos e com recursos limitados, e quanto o gestão de alto-nivel é responsável pelo processamento e analise dos eventos disponíveis no barramento de dados da rede dos pequenos dispositivos.

Foi também usado o conceito da difusão *(broadcast)* da informação, para o barramento de dados, na forma de eventos para permitir: a interligação com sistema futuros, monitorização do correcto funcionamento do sistema, captura da informação para posterior determinação do contexto; e por fim permitir a visualização do estado do sistema na interface com os utilizadores.

Por último, foi desenvolvido um algoritmo usando redes neuronais artificiais e em combinação com estatística básica que permite aprender, de uma forma autónoma, as rotinas dos habitantes em casa, conferindo a esta um ambiente inteligente. Desta forma, a casa contém o conhecimento dos padrões quotidianos dos habitantes, aumentando consequentemente o seu conforto e ao mesmo tempo, permitindo melhor eficiência energética. As análises dos dados capturados, durante dois anos completos, mostram que a redução no consumo energético pode chegar os 50%, dependendo do perfil de uso da iluminação.

INDEX

List of Figures

List of Tables

1 Introduction

1.1 Overview

Research in Ubiquitous Computing and Autonomic Computing has reached a point of convergence where the technology proliferated in the environment meets the ability of people to interact with and makes use of the possibilities that this technology creates. Advances in various fields of technology have allowed the development of devices and environments that provide computing and communication resources, that can operate in a autonomous way with ever increasing performance. The understanding of how humans interact and make use of such systems is, however, largely unresolved and often not addressed in current research. The key for the understanding of such systems and their use is the observation of humans` implicit interaction [Schmidt 1999] in context with their environments where technology is abundant.

The task of making this context information available to components in systems has become a prerequisite to advance human-computer interaction processes in Ubiquitous Computing. The context awareness and the creation of applications which use that context are the central issue of Ubiquitous Computing research. Such research raises questions on context acquisition, context representation, distribution and abstraction, as well as in programming paradigms, development support, and implications for human-computer interaction in general.

The research presented in this thesis concentrates on many of these issues like: how to acquire context in a Ubiquitous Computing environment; specify design of a system able to manage required sensors and actuators; prototype design and implementation; data acquisition and analysis.

1.2 The Ubiquitous Computing Vision

The vision of ubiquitous computing was developed by the Xerox PARC researcher, Mark Weiser, in the late eighties and early nineties. His landmark article with the title *´The Computer for the 21st Century`* [Weiser, 1991] starts off with the following sentences:

> *"The most profound technologies are those that disappear.*
> *They weave themselves into the fabric of everyday life*
> *until they are indistinguishable from it."*
> [Weiser, 1991]

Weiser's vision is one of a "computer" that disappears into the background. This does not mean that the computer visually disappears, but that it is used in a transparent way. During everyday tasks, a ubiquitous computing system would be used repeatedly without users` acknowledgement or involvement. The motivation for this vision comes from technologies that are part of our daily live, which are constantly and effortlessly used. The electricity is an example of technology that is used to power almost everything and people only think about it when no electricity is available.

Besides the notion of the computer disappearing into the environment, several other terms describe different aspects of ubiquitous computing. Weiser's vision of seamlessly integrating computers into the physical world is opposed to many of today's design principles. A seamless integration of technology into the real world would allow people to be totally unaware of technology underlying a service. Calm technology as described by Weiser [Weiser and Brown, 1995] stresses the point that good technology should not intrude on our lives, but live along with us, in a calmer way. The vision of calm technology was an inspiration for many projects in the areas of ambient and peripheral displays. Embodied virtuality is a term that was invented to explicitly bring out the ideas that computers can be brought into the physical world. This part of the vision was taken literally in the work of [Ishii and Ullmer, 1997] and in the work followed by many others in the area of tangible interfaces.

The terms context-awareness and context-aware applications were introduced a few years after Weiser's vision was published. The context-awareness is regarded as one of the main ingredients for ubiquitous computing. Context-awareness was defined in several works, most prominently by Schilit and others [Schilit et al., 1994] [Brown, 1996] [Abowd et al., 1999] [Abowd et al., 1997] [Schmidt et al., 1999]. All definitions have in common the fact that information from users or entities in the environment is acquired and used to trigger certain actions in the system. Using such context-information, systems can adapt themselves to a given situation in order to improve the users' interaction with the system. Moreover, systems can trigger actions autonomously. Finally, context can be attached to information to make it easily retrievable. The use of context information in applications needs to be embedded with the way humans live, by creating new services to the users or allowing other systems to augment its own functionality.

This close bonding between technology and everyday life is a characteristic of the ubiquitous computing vision. In order to make the vision come true, research effort is needed both in the area of technology (electronics, computer science, artificial intelligence, etc) and in human areas of research (philosophy, psychology, anthropology). Only by combining work from these areas, new insights on how to improve usability can be gained.

Today, there are some examples of how technology is integrated in everyday life showing a lack of context-awareness. The example of automatic light switches in public spaces, such as, waiting rooms, are exclusively based on a sensor which switches off after a certain period of time when no movement is detected, ignoring that people could still be in the room requiring light. Another example can be found in Swiss trains, as described by Antifakos [Antifakos et al., 2003], in which the automatic doors detect people so poorly that 70% of the people lift their hand to the sensor to trigger the door. People expect the door to open when they approach it, independently of how tall they are and how fast they are walking. Also people expect not to be bothered by an opening door when they are turning the page of a newspaper, sitting in a seat nearby a door.

1.3 Challenges on Ubiquitous Computing Vision

The ubiquitous computing vision presents several challenges that need to be overcome to make the vision a reality. Several challenges were presented by Edwards [Edwards et al., 2001] in a context of smart homes. These authors enumerated challenges that are related with hardware devices, systems integration, system overall design architecture, social behaviour and interaction with the system:

- The first challenge is the growing complexity created by a bunch of devices and systems working together. A common user needs to have the possibility to create a mental model of a system and the way devices are related, in order to understand and predict system behaviour. When a user needs to activate a function of the system, it is very important that the user is informed on several aspects of the involved data, such as, management, storage, sensitivity, privacy. However, there is a severe contradiction in this approach, as such system should be autonomous, without user intervention.

- The second challenge is the Impromptu Interoperability of devices and systems. This challenge is related with the capability of devices and systems to connect, integrate and interchange information supplied by other devices and systems in the same environment. The impromptu interoperability is even more complicated when devices are supplied by different manufactures, with different technologies. The devices and systems should work together and improve the functionalities of the system, using resources available in other devices accessible in the environment.

- The third challenge is linked to the autonomic management of the systems embedded in the environment. This challenge is related with the two previous challenges and also with the idea of a "calm technology" from the ubiquitous vision. The devices and systems have to be capable of self-management taking into account

the higher goals defined by the user. The user is not seen as administrator, but only as a non-technical person that enables the features he wants the system to perform.

- The fourth challenge is the need of the design to take into account the environment where it is installed, in terms of appropriateness to social and cultural behaviour of the users. It must also take in account the specific routine in the space where it is included. A typical example of this case is what happens when, in a home automation system, a remote user interacts with the house, shutting down all lights, not knowing that someone is having a shower. Obviously, it can just not happen. Perhaps, the remote user action can be avoided by a system design that informs the remote user that the light of the WC is switched on. This information can be enough to some users, but better context information (like someone in the shower) or the obstruction of the WC light remote control from the system could be applied in this case.

- The fifth challenge deals with the social implications of aware technologies, i.e., the technologies changes social routines in an unpredictable way. The system behaviour takes into account the users´ patterns to act, but the actions of the system can make the pattern of users also change, leading to a system re-adaptation. This can lead to an unpredictable behaviour from the users or even to a complete unsuitableness of the system to the user needs.

- The sixth challenge is the need of reliability. Very high reliability goals have to be built into the ubiquitous systems architecture itself. Although the designing for reliability requires devoting substantial time and resources that affect the system architecture, it is a culture that must be present in this type of systems. The system graceful degradation is another technology approach that must be exploited, i.e., the system must maintain some functionality level (basic functions) opposed to failing completely by a single broken component taking down the whole system.

- The seventh challenge is the inference in the presence of ambiguity. In the ubiquitous system, the decisions are based on the context to determine a particular

state, i.e., the system tries to predict the state of mind or actions of the user, but that leads to some degree of error because of the ambiguity and uncertainty data acquired from sensors in the environment. The inference of the user wanted actions, using a limited sensor input, is even difficult for the human mind and there are tasks that should not be designed to depend on machine intelligence alone.

1.4 Autonomic Computing in the Ubiquitous Computing Vision

Ubiquitous computing envisions a habitat where the abundance of devices, services and applications allow the physical and virtual worlds to become seamlessly merged. The promise of ubiquitous computing environments is not achievable unless these systems can effectively ''disappear''. In order to do that, they need to become autonomic, by managing their own evolution and configuration with minimal user intervention. It is in this context that the notion of autonomic ubiquitous computing appears, involving the facets of self-configuration and self-healing from the autonomic computing concept.

Autonomic Computing is a concept that brings together many fields of computing, with the purpose of creating computing systems that are reflexive and self-adaptive. Autonomic computing is generally considered to be a term first used by IBM in 2001 to describe computing systems that are said to be self-managing [Kephart et al., 2003]. However, the concept of self-management and adaptation in computing systems has been around for some time. The event of the combination of object-oriented programming paralleled with component-based software engineering (dynamic reconfiguration) essentially paved the way toward autonomic computing [McCann, 2003].

Blue-chip companies, like IBM, are interested in autonomic computing so as to reduce the cost and complexity of owning and operating an IT infrastructure [Pescovitz, 2002] and [Markl et al., 2003]. The aim is to allow administrators to specify high-level policies that define the goals of the autonomic system, and let the system manage itself to accomplish these goals. In these enterprises, as information systems grow larger, it is becoming increasing difficult to identify a failure in the system and repair the affected component quickly, as large systems are

heterogeneous and no single person knows the entire system. In ubiquitous systems, the same autonomic characteristics are required, i.e., there is the need of management of a large number of devices with minimum intervention by users.

1.5 Contributions

In this thesis some of the challenges of ubiquitous systems, enumerated above, are taken into account, in order to get closer to Weisers' ubiquitous vision. The major contributions are:

- A definition of a global system architecture by identifying the main blocks required by ubiquitous systems and the identification of data flow between these blocks;

- The idea that data should be radiated to the medium in opposition to data transactions based on demand requests (query/reply), i.e., the devices and modules always emit information to the medium whenever an event is triggered internally or by the environment;

- The concept of devices and systems having reflexive actions (low level reactions to events) by capturing the radiated information and immediately acting, according to local rules previously and internally defined in the system or in the devices themselves;

- Hardware devices and system components essential capabilities, on top of the obvious communication capability, that need to exist in an autonomic ubiquitous system are identified;

- Data mining methods and AI neural networks can be used to determine user activity patterns that allow construction of context awareness, which can lead to autonomic actions by the system.

1.6 Thesis Outline

The thesis is structured the following way, in chapter 2, an overview of background and related work is presented. The ubiquitous technology and hardware apparatus used to implement this type of systems are enumerated. The data flow, the methods used and the way the patterns are detected, are reviewed. The context-awareness importance is explained and the reason why it is considered to be an "Enabling Technology" is also presented.

In chapter 3, the main components and system structure of an autonomic ubiquitous system are presented. The requisites to create an autonomic management system are enumerated. The characteristics as well as the solutions used to manage a distributed system are presented, which are the guidelines to the overall system architecture philosophy.

Chapter 4 focuses on the hardware -sensors, actuators and communicators- that enable the vision of ubiquitous system. It is shown how a sensor can be inserted in the environment, how data are captured and finally transmitted. The sensor data are assessed and classified. The actuators are enumerated, classified and their management functions are presented. The communicators are presented as a facilitator in the construction of the system.

The Chapter 5 shows how the management system can be built, in a low-level or local perspective, using the previously mentioned hardware characteristics. The autonomic management system is addressed at hardware level, i.e, how the system behaviour can be embedded in devices existent in the environment. The approach has implications in devices firmware and system network architecture, in order to achieve high robustness with graceful degradation of the whole system.

In chapter 6, a case study is presented, using the system architecture suggested in this thesis. The case study consists of the instrumentation of a home in a non-controlled environment, i.e., the study of a home in a real life family scenario.

The building of the context-awareness is discussed on chapter 7. The context construction uses data mining techniques, going from basic data statistics to more complex AI neural networks techniques. The output usage, as well as the user pattern and behaviour, is analyzed in order to make system autonomic actions possible.

On chapter 8, the data acquired in this project is analyzed and the results are assessed, allowing the validation of the adopted system architecture, by careful analysis of its advantages and drawbacks.

Finally, the chapter 9 summarizes the contributions presented in the thesis, listing what was achieved and what is reserved as a future work. The implications of this type of technology, some critics and dangers related with it are also discussed.

2 Ubiquitous Computing Related Work

In this chapter an overview of related work is presented, projects and groups that are investigating ubiquitous systems are investigated. The ubiquitous technology and hardware apparatus used to implement this type of systems is studied. The data flow and methods used as well as the way patterns are detected are also reviewed. Context-awareness importance is explained and the reasons why it is considered an "Enabling Technology" are presented as well.

2.1 Projects

AWAREHOME

The AwareHome[1] Research Initiative, from Georgia Institute of Technology, intended to create a living laboratory for ubiquitous computing research on everyday activities [Kidd et al. 1999]. This project started in 1998, constructing two identical living spaces that include a kitchen, dining room, living room, two bedrooms, two bathrooms, one office and laundry room, in more that 468 m^2 home that operates as a living laboratory for interdisciplinary design, development and evaluation. Commercial off-the-shelf (COTS) and state-of-the art technologies are used to instrument the house and to create ubiquitous applications.

This initiative is devoted to the multidisciplinary exploration of emerging technologies and services for the home. The research areas in the AwareHome are: chronic care management in the home; future tools for the home; and digital entertainment and media. Some examples of technology developed on this project are: the smart floor [Robert et al. 2000], where pressure sensors are placed in the floor of the house in order to detect footstep force profiles allowing the user's identification in a transparent way; Powerline Positioning [Patel et al. 2006] uses the power line of the house to irradiate a signal injected by a standard device, allowing the positioning

[1] Available on-line at: http://awarehome.imtc.gatech.edu/

determination by the signal reception strength. These group focus on the sensing components and algorithms to process the sensor data but they are not focusing on the autonomic intelligence of the architecture.

Lately, in the technology area, this group started to study the home network problem of setup and management as presented by Shehan and Yang [Shehan et al. 2007] [Yang et al. 2007]. This work points out that better Human–computer interaction (HCI) is required by users for a better management of the systems at home.

MavHOME – Smart Home project

The MavHome[1] Smart Home project [Cook et al., 2003] is a multi-disciplinary research project at the Washington State University and the University of Texas at Arlington focused on the creation of an intelligent home environment. This project approach is to look at the smart home as an intelligent agent that perceives its environment through the use of sensors, and can act upon the environment through the use of actuators. The home has certain overall goals, such as minimizing the cost of maintaining the home and maximizing the comfort of its inhabitants. In order to meet these goals, the house must be able to reason about and adapt to provided information. The multi-layer architecture consists of physical, communication, information and decision layers that are based on CORBA[2] for remote method invocation.

The MavHome is equipped with X10[3] ActiveHome kit and HomeSeer[1], thus allowing the inhabitant to automatically control the appliances. The authors used motion-sensors placed along

---

[1] Managing and Adaptive Versatile Home (MavHome) Available on-line at: http://ailab.wsu.edu/mavhome/index.html

[2] The Common Object Requesting Broker Architecture (CORBA) is a standard that enables software components written in multiple computer languages and running on multiple computers to work together. This is a mechanism in software for normalizing the method-call semantics between application objects that reside either in the same address space (application) or remote address space (on a network remote or local host). More information available on-line at http://en.wikipedia.org/wiki/CORBA

[3] X10 Powerline Carrier (PLC) Technology X10 Home Automation homepage. Technical information available online at http://www.x10.com/support/technology1.htm

the inhabitant's routes, in the home, and with that information try to predict the location of users to improve energy savings and increase inhabitant's comfort. The Mavhome network is divided by zones and with a centralized server (main cluster) that processes all the information. The zone topology helps in the location and association in specific user tasks [Roy et al. 2003].

The NavHome research achieved good results on predicting the user's actions but because of the technology used and the centralized processing of data the intelligence acquired cannot be embedded in the environment in an autonomic way.

PlaceLab

The PlaceLab[2] is a real home where the routine activities and interactions of everyday home life can be observed, recorded for later analysis, and experimentally manipulated. Volunteer research participants individually live in the PlaceLab for days or weeks, treating it as a temporary home and at same time, in background, a detailed description of their activities is recorded by various sensor devices integrated into the labs architecture. PlaceLab facility is a 93 m^2 foot lab consists of a living room, dining area, kitchen, small office, bedroom, full bath and half bath. The interior of the PlaceLab is formed by 15 prebuilt and reconfigurable cabinetry components. Each contains a microcontroller, an addressable speaker system, and a network of 25 to 30 sensors. New sensors can be easily added to this network as required. Existing sensors record a complete audio-visual record of activity. All sensing devices are discreetly integrated into the cabinetry, appliances, and furniture and fixtures [Intille et al., 2004]. Interior conditions of the apartment are captured using distributed sensors to measure: temperature (34 sensors), humidity (10 sensors), light (5 sensors), and barometric pressure (1 sensors). The PlaceLab also features electrical current sensors (37 sensors), water flow (11 sensors) and gas flow (2 sensors). Small wireless sensors that detect movement can be also be taped onto any movable objects or worn by the participant with wrist bands or ankle bands [Intille et al., 2003], [Tapia et al, 2004a].

---

[1] HomeSeer is a home automation software and pack of hardware solutions. More information available on-line at: http://www.homeseer.com/

[2] Available on-line at http://architecture.mit.edu/house_n/placelab.html

The PlaceLab project shows, in practice, that creating an environment that monitors the human activity with a use of numerous sensors can be a technical challenge using current technology. This is especial noticed in the server "closet", with 3,7 m^2 area, where all the cables from the home converged to a amazing total of 20 personal computers [Intille et al., 2006]. The complexity of the whole system was such that researchers create a special link to a researcher infrastructure to allow the remote monitoring of the devices functionality, and this way, fixing as fast as possible any malfunction that may happen. The autonomic management embedded in the infrastructure of the ubiquitous systems helps the construction of this type of environments minimizing the maintenance problems.

Project Aura

The AURA[1] project at Carnegie Mellon University aims to build a "distraction free", i.e., unobtrusive, ubiquitous computing environment [Garlan et al., 2002]. It is an umbrella project with different subprojects collected under its central goal, including projects like the Coda file system and Odyssey (Remote monitoring and adaptation) [Satyanarayanan, 1996]. Proactive adaptation of applications has also been explored by Judd [Judd et al., 2003], which describes Contextual Information Service (CIS). The CIS is a lightweight interface to obtain context information via virtual databases.

It has been implemented a concept of task-driven computing by capturing user intent and mapping it into a task corresponding to a set of abstract services. Additionally, the services are achieved by the environment infrastructure providing continuous support to user tasks regardless of the environment in which the user is in. The "Aura" of each user represents the set of services required to accomplish a task or activity, allowing the user to move from environment to environment while keeping the task in execution with the resources available in that environment.

The system implementation uses standard computing and networking technology and therefore presenting the consequent limitations.

---

[1] Available on-line at http://www.cs.cmu.edu/~aura/

Gaia

Gaia[1] project, from Digital Computer Lab at University of Illinois, tries to expand the traditional operating systems to the ubiquitous computing system, and with this simplifying the implementation of application and management of the spaces [Roman et al., 2002]. The main contribution of Gaia is not in the individual services, but instead, in the interaction of these services. This interaction allows users and developers to abstract ubiquitous computing environments as a single reactive and programmable entity instead of a collection of heterogeneous individual devices. Gaia provides mechanisms to allow users to configure their applications to benefit from the resources contained in their current space. Users can interact with multiple devices simultaneously, can reconfigure applications dynamically, can suspend and resume groups of applications, and can program the behaviour of applications based on context attributes. Gaia emphasizes the interaction between users and active spaces.

The Gaia OS provide some mechanism to create some fault tolerance in the system. This mechanism works at application level, i.e., it saves periodically the state of the running application. Each application sends a periodic heartbeat message to the Gaia OS informing that it is alive. When an application fails it terminates and no longer sends heartbeat messages. [Chetan et. al., 2005]. If the timeout is reached, the Gaia OS it notifies the fault manager. The fault manager obtains information about the current context from the context infrastructure and it gets device and application properties from the Space Repository. It uses this information to infer a contextually appropriate surrogate device on which the application can be restarted. The failed application is then restarted on the alternative device using the saved state from the checkpoint storage.

[1] Available on-line at http://gaia.cs.uiuc.edu/

Equator

The Equator[1] Interdisciplinary Research was a six-year research project collaboration between eight academic institutions in the UK, funded by the U.K.'s Engineering and Physical Sciences Research Council. It involved over 60 researchers, with a range of expertise that encompasses computer science, psychology, sociology, design and the arts. Equator pursued research on how digital and physical realities can be mixed in everyday life. They track a portfolio of projects ranging from urban games to digital care, and from museum visits to domestic technologies, as a way of exploring fundamental research challenges involving interaction, infrastructures and devices. A publicly available framework, called Equip, for C++ and Java supports a shared data service for inter-application communication and event handling and was used in a few applications within the project [Greenhalgh et. al., 2001].

The Equator include projects like VoodooIO that allows the creation of input interfaces configurable and open to users objectives [Villar et al, 2007]. In this project users can position in the environment, among other things, slides, joysticks, buttons, creating new control layouts. Controls are placed in a substrate that contain two conductive layers that supply the controls with relevant information via a coaxial pin connector and using a 1-wire[2] protocol. Another project, Drift Table [Gaver et al., 2004], which is an electronic coffee table that displays slowly moving aerial photography controlled by the distribution of weight on its surface. It was designed to investigate ideas about how technologies for the home could support ludic activities, i.e., activities motivated by curiosity, exploration, and reflection rather than externally defined tasks. This project brings new functionality to every day object showing that the ubiquitous technology does not need to try only to make things for users but rather allowing the construction and living new experiences.

[1] Available on-line at http://www.equator.ac.uk/

[2] 1-Wire provides low-speed data, signalling and power over a single signal, using two wires, one for ground, one for power and data. More on-line in: http://en.wikipedia.org/wiki/1-Wire

TEA – Technology for Enabling Awareness

The Technology for Enabling Awareness (TEA)[1] project was a joint project between Starlab[2] , TecO at the University of Karlsruhe in Germany, Omega Generation at the University of Bologna in Italy and Nokia Research in Finland. Its aim was research on context awareness with handheld and wearable devices with a focus on low-cost sensors and mobile phones [Chen et al., 1999]. The project has developed a hardware board for collecting sensor data and connected it to a standard mobile phone. The adaptive context awareness was achieved by the use of machine learning algorithms on low-level sensor data; those algorithms should adapt to the most frequent user contexts. One of the project goals was also unobtrusiveness, i.e. to demand as little human attention as possible.

The TEA project confirmed the need to pre-process the sensor data to get two-step perception approach with feature extraction prior to fusion. The research also points out that there are some sensors that contribute more to the context detection than others, e.g., they find that the temperature or pressure are not so important as light, audio and acceleration to context detection [Gellersen et al, 2000]. These conclusions may be explained by the large constant time of temperature and pressure in contrast with other mentioned sensors in the pursued context-aware application. Another conclusion drawn is the location of a wearable sensor, which can influence the system performance. This problem is a major issue of the ubiquitous system implementation that affects users to wear some type of devices, RFIDs, badges, etc. Users can wear devices in a inadequate place or simply not wear them at all, resulting in an inevitable system failure.

MEDIACUP

The Mediacup[3] project approaches the implementation of a ubiquitous system by augmenting every day objects with technology that can sense and transmit its own context. Researchers from TecO at the University of Karlsruhe in Germany created a coffee cup that can sense its temperature

[1] Available on-line at http://www.teco.edu/tea/

[2] Official site at http://starlab.es/

[3] Available on-line at http://mediacup.teco.edu/

and movement, and transmission of that information to the environment by infrared light [Beigl et al., 2001]. The information transmitted can be used to create context services but, and at same time, and because of the nature of the transmitted medium, the information can be used by other devices. It has also been created the HotClock that receives information directly from the cup and avert the user that is manipulating the coffee cup if the liquid is too hot to drink.

The concept of broadcasting information in the environment is also discussed in the thesis but in a global system architecture view, allowing the impromptu interoperability of ubiquitous devices and systems.

The Mediacup project implementation also takes important practical restrictions into account, e.g., very low power consumption, wireless recharging. This aspect is essential to wireless devices in a ubiquitous environment as it affects the robustness and the usability of this type of systems, i.e., the system can not constantly annoy the user to change the batteries of the devices which inevitably lead to devices not working, resulting in bad system performance on a real situation.

Smart-Its

The project Smart-Its[1] merges the experience gathered with the MediaCup and the TEA projects, focusing their research from context awareness in single devices to ad-hoc networking of context-aware devices [Gellersen et al., 2002]. Smart-Its, is an experimental hardware platform, integrating, sensing, processing and communicating capabilities into a single, small device. Its design is modular, separating the sensor unit from the core unit, which provides processing and communication. With Smart-Its, ad-hoc networking, sharing of context information and sensor fusion across devices is investigated. It uses Bluetooth for communication to ensure interoperability with consumer devices. Issues like the minimization of physical size and power consumption were taken into account in order to support rapid prototyping.

[1] Available on-line at http://www.smart-its.org/

Sentinel Computing

The Sentient Computing project[1], from AT&T Laboratories in Cambridge and University of Cambridge, seeks to combine sensors and computers to monitor resources, maintaining a computational model of the world, and acting appropriately [Addlesee et al., 2001]. Researchers use a building-wide ultrasonic location system with mobile devices called "bats" and a full covering space model of the building. Using information extracted from this location system (user identity, location and nearby persons and objects), documents are tagged with meta-data to ease a later context-aware information retrieval. As a building-wide project, it can rely on powerful infrastructure support like database systems for storing context information [Harter et al., 2002].

The “bat” device presents a very good indoor location with resolution of about 3 cm which is good enough to allow it to be used as an input device. Users can push a button on the device and at same time place it over a sticker label on the wall, to do a specific action. The constant usage of the device minimizes the problems of users not wearing it because users need the tool to carry out their daily work.

ACCORD

The ACCORD[2] (Administering Connected Co-Operative Residential Domains) project, from a consortium of Swedish Institute of Computer Science AB (SICS) and Acreo from Sweden and University of Nottingham from UK, has explored how everyday life can be supported and enhanced through the use of collections of interacting artefacts. The main aim of the ACCORD project is to develop facilities to construct, administer and manage future interactive home environments by the arrangement of interactive devices as the principle means of controlling the complexity inherent in domestic environments. In order to achieve that objective the ACCORD developed the Tangible Toolbox, based on a shared Data Space that enables people to easily administer and re-configure services based on embedded devices around the home [Åkesson et al., 2002]. This toolkit also

[1] Available on-line at http://www.cl.cam.ac.uk/research/dtg/attarchive/spirit/

[2] Available on-line at http://www.sics.se/accord/

enables devices to be integrated with each other through several and different "composition editors". Those editors allow users to connect device and actions in a puzzle like schema.

The approach of the ACCORD project tries to solve the implementation and management of a home ubiquitous system by allowing the user to create sequences from sensors to actions by putting together blocks as a puzzle [Rodden et al., 2004a]. This approach works very well if the available blocks are relatively in small number, as it is possible to even create a large number of combinations, but if the number of available blocks increases the users have difficulty to find the correct block or understand the difference between them. Despite these limitations, if the blocks represent context-situations and possible actions, the method is a convenient solution for reconfiguration of the systems at home [Rodden et al., 2004b].

2WEAR

The 2WEAR[1] project explored the concept of a personal system that is formed by putting together computing elements in an ad-hoc fashion using short-range radio. Certain elements are embedded into wearable objects, such as a wristwatch and small general-purpose compute/storage modules that can be attached to clothes or placed inside a wallet. Others have the form of more conventional portable computers, like PDAs and mobile phones. Also, there are stationary elements as part of the environment, some of which are visible, such as big screens and home appliances, while others are not directly perceivable by the user, such as network gateways and backend servers [Lalis et al., 2003].

Interaction between the various devices of the system is based on the concept of services. A service denotes a hardware and/or software resource modelled as a distinct unit of functionality that can be accessed over the network. The implementation of the communication protocol is on top of Bluetooth L2CAP[2].

---

[1] Available on-line at http://2wear.ics.forth.gr/

[2] The Bluetooth Logical Link Control and Adaptation Layer Protocol (L2CAP) is layered over the Baseband Protocol and resides in the data link layer. L2CAP provides connection-oriented and connectionless data services to upper layer

2.2 Pioneer Projects

In this section, the most important and pioneering projects that created and influenced the development of subsequent researches in ubiquitous systems are presented. These describe an important part of the history on ubiquitous computing and help to understand how research advanced.

Active Badge

The Active Badge[1] system is a wearable, personal badge equipped with a microprocessor, an infrared emitter and a button. Every 10 seconds, it sends a beacon containing its unique id to receivers distributed in the environment, allowing gathering location information at a central server. It uses infrared as primary communication medium, which can be embedded in small devices with limited battery life [Want et al., 1992]. These badges have been used at Olivetti Research Ltd. (ORL) for pioneering applications in pervasive computing, like a building-wide notification system, and have since then been deployed at the University of Kent, Imperial College, London, Lancaster University, the University of Twente, Xerox PARC, DEC research laboratories, Bellcore and MIT Media Lab.

This research stimulated the development of the "bats" device which used ultrasonic, capable of precise location detection, used in the sentinel research project.

---

protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

[1] Available on-line at http://www.cl.cam.ac.uk/research/dtg/attarchive/ab.html

Xerox ParcTab

Inspired by the Weisers' vision described in [Weiser, 1991] the Xerox Palo Alto Research Center (PARC) developed the ParcTab[1] system. It consists in a palm-sized PDA with touch screen, complemented by an infrared communication infrastructure with room-sized cells. Most of its applications are executed on remote hosts and thus depend on the communication infrastructure, which also handles location tracking. One of the applications that have been implemented on the ParcTab is a remote control system to control lights and temperature for the current location; others include the better known Forget-me-not system [Lamming et al., 1994]. The ParcTab can be seen as a more complicated version of the Active Badge that includes a limited user interface for arbitrary applications.

The Adaptive House

Learning habits of users has previously been explored by Michael C. Mozer in The Neural Network House [Mozer et al., 1995], later named as The Adaptive House, which is able to predict occupancy of rooms, hot water usage and likelihood that a zone is entered in the next few seconds using trained feed-forward neural networks [Mozer, 1998]. It used motion detectors and X10 devices to capture and act on the house environment. The context information was again mainly comprised of location, but additional state information from rooms like the status of lights or the temperature set by inhabitants were used. While this project is one of many "smart house" projects, it was one of the first to include prediction of user actions using a feed-forward multi-layer perception Artificial Neural Network (ANN), trained with standard back-propagation [Mozer, 1999].

The author of this project identified proactivity as the main problem of this type of systems. It pointed out that incorrect autonomic actions inevitably frustrate users to a point of having to completely remove that functionality [Mozer, 2005].

[1] Available on-line at http://sandbox.xerox.com/parctab/

2.3 Ubiquitous Systems Concerns

The projects and research presented in this chapter show a diversity of approaches to try to build a ubiquitous system. Some projects focus mainly on the acquisition of information about users activities; others try to help users on standard activities by enabling the execution of tasks in different platforms and places; some let users create new and personalized input layouts in order to adapt to user needs; some let users define rules of what the system should do in a specific situation.

2.3.1 Context-awareness

All the projects mentioned in this chapter have the context-awareness as a system objective in order to achieve the implementation of the ubiquitous systems. Context-awareness allows systems to predict user's objectives or situations and therefore prepare applications, interfaces and spaces to consequent user's tasks.

In a ubiquitous system, the system must help the user by supplying what is important for that particular moment.

> *"Machines that fit the human environment instead of forcing humans to enter theirs will make using a computer as refreshing as a walk in the woods"*
>
> [Weiser, 1991]

The statement from Weiser points out that systems must be used effortless, i.e., must be used in a natural way. To achieve this, the system needs to known, in some degree, what the user needs, i.e., the context of the moment. This means that the system needs to gather information of What, Who, Where, When, and How to build the context that can effectively help users.

2.3.2 Hardware Apparatus

In order to acquire context, the projects previously described try to acquire the information from a range of sensor devices or from the interaction of users with the systems. Some projects use COTS sensor devices, other projects develop custom made hardware sensor solutions. The usage of COTS sensor devices provides off-the-shelf components for a testing system, however because technologies involved are not dedicated to the construction of ubiquitous systems, they require a special attention on setting up and maintenance. Another limitation of the usage of COTS devices is their closed design, not allowing changes in its firmware limiting the researcher to use the SDK made public by manufactures.

The specific devices implemented on some projects try to acquire information that standard devices normally do not supply. The indoor location is one of the context attributes that are often chosen by researchers because these devices can respond to “who” and “where” for context building. The solutions normally adopted require the usage by the user of a wearable device with a unique ID. The identification of users is carried out using a table that matches a person to a specific device ID. Wearable devices allow the construction of a truly ubiquitous system because the infrastructure is always “attached” to the user. However, wearable devices have a fundamental problem which may cause system failure, i.e., if the user does not wear the devices, the system cannot interact with the user at all. This does not mean that wearable device should not be used in this system, but they should be used taking into account this limitation.

The implicit gathering of information by the surrounding environment devices is an approach that fits better in the Weiser’s Vision for the construction of a ubiquitous system. In this approach, the user does not need to engage the attention to any specific devices and the system can capture information of the specific context to change the potential environment response. Some projects previously presented show some devices that use this approach.

2.3.3 Data Flow and Structure

In the implementation of the ubiquitous systems presented, data flow in the systems tends to be managed in a centralized way, i.e., data captured by sensors are filtered, then sensor fusion is applied and finally the context is obtained in a centralized processing unit. The actions are then coordinated by a central server that received all the information and have all the high level roles that rule the system behaviour.

A centralized structure can be used to implement this type of systems as it has all the necessary information available for decision making. Nevertheless, a centralized system presents also vulnerabilities in terms of robustness, fault tolerance, latency and scalability. The system is dependent on a centralized unit to execute its functions, high level and basic ones, and if the centralized processing unit or data network fails the system stops responding. Also, because a centralized processing unit needs to execute all functions of the system, the complexity of that device is greater and therefore more prone to software "bugs" and crashes. Related with the complexity, any down time necessary for rebooting or maintenance has a greater negative impact on the system. In the centralized structure even if the sensor and actuators are available the absence of a centralized coordination makes them useless. As a system increases in size the limitation of a centralized structure can also start to become a problem as the latency of the system may become a serious problem.

A distributed structure solution seems to be more adequate for the implementation of a ubiquitous system. In this system it is expected that devices work together sharing information. In a distributed structure, it does not exist a centralized point of failure, resulting in a more robust system. The scalability and latency is also an advantage of a distributed structure. But a distributed structure also has some problems like debugging, resulting in increased development time, the impromptu interoperability between devices and the difficult of definition of global objectives, as the vision of the whole system is much more complex.

It seems that the most appropriate solution for implementing ubiquitous systems is a hybrid system structure solution, i.e., centralize some functions that need more information and are more demanding and, at same time, embed critical functionalities in the devices spread in the

environment. This approach requires the classification of all functions of the system, allowing the identification of the core functions to embed in the devices and functions that are less critical where failure has less impact in whole system. This structure helps to divide the system in smaller and simple sub-system that work independently, reducing at the same time the complexity of the main centralized processing unit.

2.3.4 Proactivity

Proactivity, as a real anticipated action to the user executed by the system, seems to be avoided in the implementations of the relevant projects in the field mentioned above. It may be, because those implementations present such a high annoying failure rate that invalidates its usage. The proactivity expected from this type of system are automatic actions that are executed in advance avoiding user intervention. The problem rises when an action performed by the system is wrong. In this case it could have the potential to be unpleasant or even dangerous. Nevertheless, proactivity can and is used in situations where an erroneous action is not so critical, e.g., the case of a message being relayed to a screen of a PDA giving some information that it is completely out of context. In this situation the user can simply choose to ignore such message automatically generated by the system.

2.4 Summary and Conclusions

In this chapter some projects, groups and initiatives were presented which represent the most notorious projects on the ubiquitous computing systems field.

The home environment is a scenario chosen by various projects as it is an ideal place to test ubiquitous systems. This is the case of the AWAREHOME that focus on everyday activities in home environment, the House_n/PlaceLab that tries to access the routine activities and interactions of everyday home, the NavHOME that aims the creation of an intelligent home environment; the Adaptive House that implemented prediction of some devices usage inside a home. Further, the

aim of the ACCORD project has to develop facilities to construct, administer and manage future interactive home environments.

In a more global interaction with real active environment and users, other projects are: AURA that aims the construction of unobtrusive ubiquitous computing environment; Gaia that emphasizes the interaction between users and active spaces; EQUATOR that points out how physical realities can be mixed in everyday life.

The wearable and mobile devices were also analysed in order to help the implementation of a ubiquitous system. The problems of putting together wireless computing elements in an ad-hoc fashion using short-range radio were discussed in the project 2WEAR. Also related, the MediaCup and Smart-its used every day objects, enhanced with technology that can sense and transmit its own context, using ad-hoc networking.

The ubiquitous computing environments are characterized by an environment filled by devices working together, sharing information and services in order to help users. This characteristic leads, inevitably, to high complex systems, which collides with the unobtrusive nature wanted in this sort of systems. To address this fundamental problem it is fundamental to add some degree of autonomy of the devices embedded in the environment.

The ubiquitous system with autonomic management can be called autonomic ubiquitous system and is discussed in the next chapter. The requirements to create an autonomic management system are discussed. The characteristics as well as the solutions used to manage a distributed system are presented in this project, providing the guidelines of the overall system architecture and philosophy proposed in this thesis.

3 Autonomic Ubiquitous System in Home Environment

In this chapter, the main components and system structure of an autonomic ubiquitous system are presented. The requirements to create an autonomic management system are described. The characteristics as well as the solutions used to manage a distributed system are presented, having in mind the overall system architecture philosophy of the proposed Autonomic Ubiquitous System.

3.1 Ubiquitous Computing with Autonomic Computing

The ubiquitous computing systems vision proposes the creation of applications that react to a situation, learning the behaviour of the users and use it to create intelligent applications that can assist passively or actively in everyday tasks. The motivation for this vision, formed by Mark Weiser, is to create the usage of technology effortless - [Weiser, 1991], [Weiser, 1993a] and [Weiser, 1993b]. The system purpose is to free the user from the details of the technology, which was named the “Calm technology” as described by Weiser [Weiser and Brown, 1995] where the technology should not intrude on our lives, but live along with us, in a calmer fashion. To achieve this, the technology should be invisible but everywhere and embedded in the environment.

Autonomic computing provides the ability of a ubiquitous system to effectively "disappear". It is a concept that brings together many fields of computing, with the purpose of creating computing systems that are reflexive and self-adaptive, i.e., the systems works autonomically (without the interaction of user or administrator) reacting to situations, changing and adapting its behaviour.

3.2 System Overview

The autonomic system needs to implement the goals of self-government and self-adaptation, which reflect the capabilities of self-configuration, self-healing, self-optimization and self-protection. These capabilities allow the measurement of the autonomic system fitness to implement the proposed ubiquitous system [McCann et al., 2004].

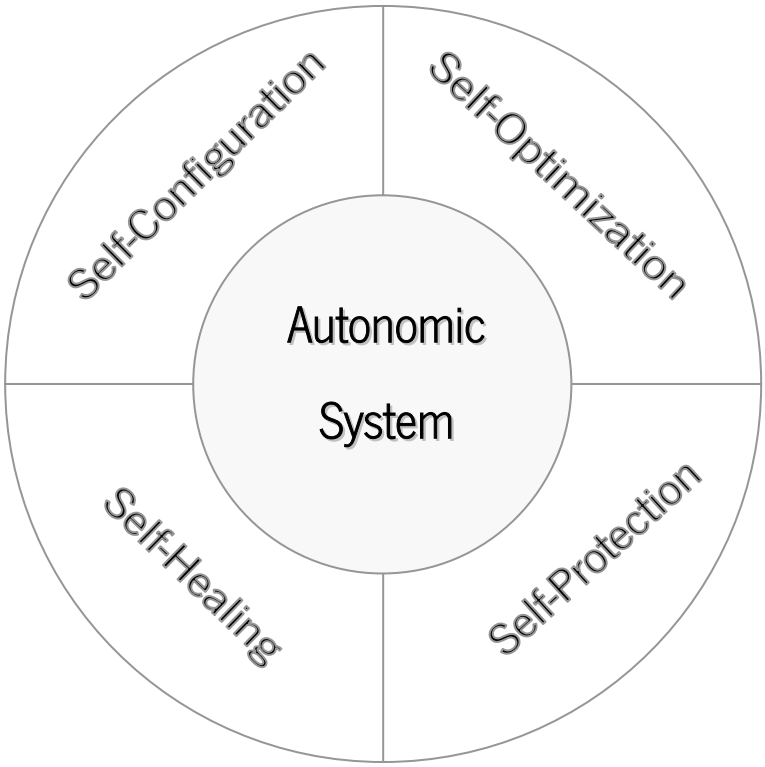


Figure 1 – Autonomic System Capabilities

These capabilities can be analysed from two distinct system perspectives, a global view and a local view: where the global view is concerned with high level services and analyses and control; in opposition to the local view perspective that is more technology-dependent (hardware characteristics), controlling hardware devices and network.

3.2.1 Global View Perspective

The global view perspective is concerned with maintaining a high level functionality and operation of the whole system. From a global view perspective the autonomic system capabilities are:

- self-configuration that is achieved by defining high level goals, i.e., by specifying what is considered necessary to be present on a required function. This means that it needs to be able to install itself, based on what is available in the infrastructure that can be used for that particular function. As an example, a higher goal can be the detection of human presence in a particular room. The system self-configuration is the ability to find the patterns in the data supplied by room embedded devices in order to achieve that higher goal. The information can be supplied by a motion detector or any other source that is related to activity in a room and the system must be able to use it;

- self-healing is reached by retrieving the required information from other sources, and still be able to operate even if it loses some of its effectiveness. In the example above the healing is the ability of the system to continue to detect the human presence in the room even when the motion detector fails. The system, for example, can use light activity in order to carry on its normal operation;

- self-optimization is accomplished by constantly trying to get better and more reliable information. This capability is similar to self-healing, differing in the fact that it wants to improve the actions of the functions, in opposition to the self-healing that tries to guarantee the operation of the function. In order to achieve this, it uses the high level goals defined in the self-configuration. Still using the same example mentioned above, the system selects the information that comes from the motion detector because it is the most reliable source of activity in that particular room;

- self-protection which is necessary to assure that the information used is correct and to verify the implication of the function results, i.e., the self-protection is the

"conscience" of the whole system. This capability ensures that the information supplied by devices is consistent, enabling or disabling accordingly the usage of a particular functionality.

Users, in this perspective, are part of the main loop, by providing implicit information that the system tries to understand and consequently causing, an adaptation of its functions.

Figure 2 shows the capabilities that users employ when they interact with a particular environment. Users have the knowledge of what is needed in order to achieve an effect in the environment. The resulting action depends on the knowledge of the users and the high level goals established in the system. To better illustrate this interaction it can be imagined, as an example, the switching of lights in a home environment and the system high level goal of automatically controlling it. Users know, *a priori*, that they need to touch a switch to change the light state. The system implicitly monitors the light usage and tries to extract patterns of the habits of the users taking into consideration the usage of the lights. When the system determines patterns, with some degree of precision, it can start to anticipate the actions of the users, as defined in the high level goals, and then the self-configuration can be achieved.

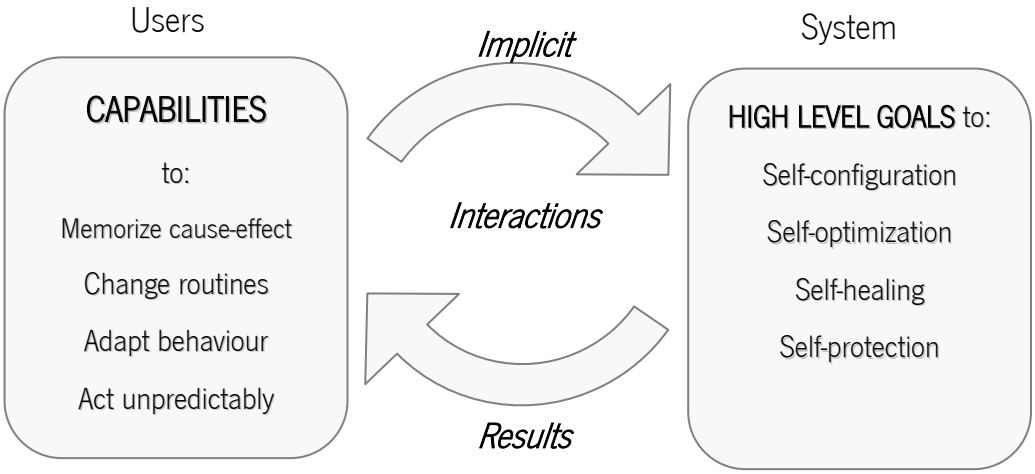


Figure 2 – Global interactions loop between users and the system

The detected action patterns of the users by the system inevitably change over time, so the system needs to continuously monitor and update its configurations. The constant stream of data supplied by the actions of the users also allows the system to improve the pattern determination and associated precision. This process enables the self-optimization of the system.

When any component of the system fails, the user adapts his behaviour. This does not mean that the user knows how to repair or replace the failing component. The adapted behaviour is a normal response from the user to a new situation. This new pattern from the user can show a considerable difference from previous detected patterns. The system detects this new pattern and updates its reactions. This process can be seen as a self-healing capability from the system. In the example exposed earlier, it can imagined a failure in a light bulb. The user notices that the light was not automatically turned ON and the user tries to switch it ON. Because of the failure, the user is incapable of performing that action and chooses to switch another light in the room. This new action is captured by the system which adapts to this new situation.

The self-protection on this global view perspective consists in blocking automatic actions when the detected user patterns appear not suitable to given situations. This can be detected when a user executes a completely opposite action immediately after an automatic action from the system. The system backs-off when that interaction has happened, by increasing the threshold for the activation of a detected pattern. If the increment is sufficiently high, the automatic function can be completely disabled.

3.2.2 Local View Perspective

In an autonomic system, the local view perspective objective is to preserve the flow of information and it is not concerned on the functions that use that information. This allows the best quality data to be always available. This focuses on the way the network and specific devices work and therefore its capabilities depend on the technology used in a specific implementation.

The self-configuration, in this perspective, aims the autonomic setup of the network parameters. These parameters include the automatic setting of the network address, the

auto-configurations of the medium access or restoring the configurations when replacing a faulty device. The self-healing solves the problem related with the routing of the data when the connection is lost, e.g., selection of an alternative gateway. The self-optimization tries to optimize the flow of the information, e.g., by choosing a better quality data channel or change the data transfer rate. The self-protection checks if peer devices have proper access to get, receive and verify the integrity of data. It is also responsible for blocking the access to the data bus of a malfunctioning device that is flooding the network.

In order to allow the autonomic behaviour from a local view perspective point of view, some characteristics must be present in these low level devices. These features are presented in the following chapters.

3.3 Middleware Block Diagram

An autonomic system needs to constantly monitor the environment, in order to be able to adapt its behaviour. In Figure 3, it is presented a block diagram of an autonomic ubiquitous system. In the diagram, the system is divided into five blocks with specific functions that allow the implementation of the global and local view perspectives and finally achieve the automatic behaviour. The five blocks are: Sensors; Actuators, Data & Hardware Management, Context Services and Applications. The Sensors and Actuators provide events that are stored in the Events Database. The Events Database raw data is used by the context services to compute the context awareness that is accessed by the applications using the API. If an application requires an action to the taken the context services are responsible for its translation to a specific action command. The command is then relayed to the actuator by the Action Coordinator. The Action Coordinator is part of the Data & Hardware Management and it operates as an action gateway between the hardware and the high level software.

As the local view perspective is more hardware-dependent, it only uses sensors, actuators and Data & Hardware Management blocks. In the global view perspective, the context services and application blocks represent the higher level and more complex functionality.

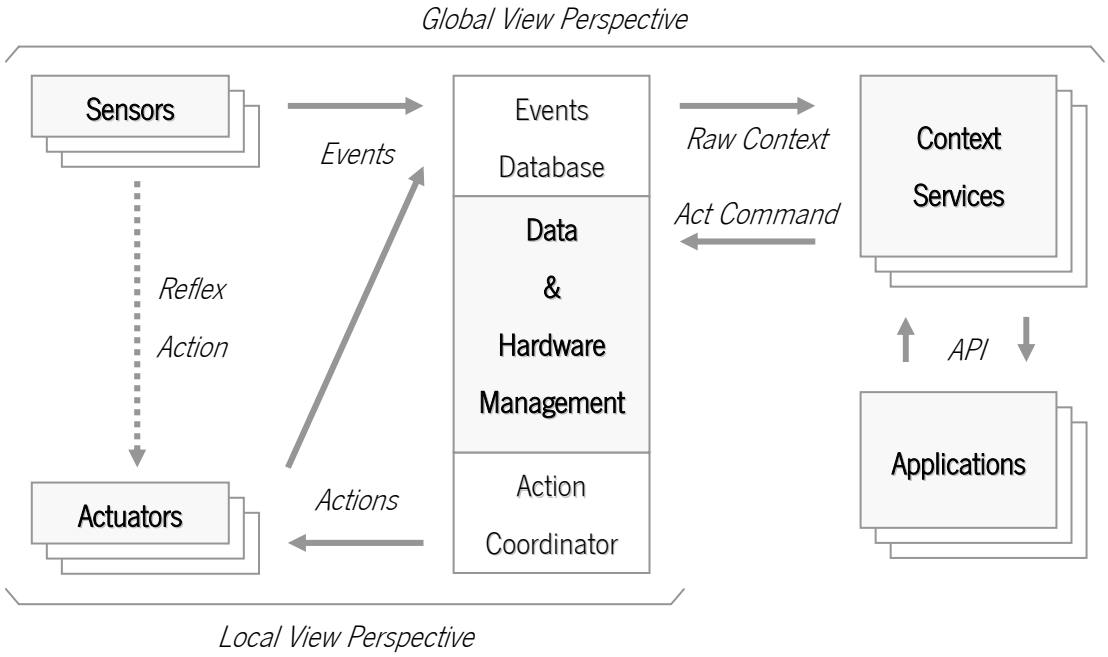


Figure 3 – Block diagram of an autonomic ubiquitous system

3.3.1 Sensors Block

The sensors block represents all kinds of devices that can sense the environment. These sensors can be divided into two classes in terms of the type of sensed data, as shown in Table 1. The *Environment Variable Sensors* are sensor devices that can measure an environment variable in a continuous sequence of values, e.g., temperature, light, etc. The *Environment Action Sensors* are sensor devices that can detect activities or events that happen in the environment. These sensor devices are activity-focused, but sometimes they also supply its state, e.g., the light switch can be ON or OFF. The examples of action sensors that only supply the event, not providing the state, are: Glass Brake Detector, Fingerprint, etc.

Table 1 – Environment sensor classes (some examples)

Environment Variable Sensors	Environment Action Sensors
• Temperature • Light • Moister • Noise • Weight • Audio Noise level • Air quality • among others	• Light Switch • Door Opening Detector • Passive-Infrared Detector (PIR) • Glass Brake Detector • Floor switch • Fingerprint • RFID • among others

The information of the sensor is communicated to the upper level layers of the system in an event-driven system. In the *environment action sensors*, the sending of information is straightforward, because this class of sensor is event-oriented. But in the case of the *environment variable sensors,* the events need to be generated to allow them to be sent to the upper layer. This can be achieved by defining a refresh interval to send the data or defining an alarm generation schema for the variation of the data, i.e., when the sensor data changes more than a value defined in the threshold, an event is generated. The information provided by the *environment action sensors* is more reliable and precise than the one in the *environment variable sensors* and consequently it facilitates the creation of context awareness.

The information retrieved from the environment by the sensors block is stored in a database of events. A time stamp is added to every event to allow a later offline context-awareness processing. The database can be distributed and stored in the sensor devices in the form of a rollover logger.

3.3.2 Actuators Block

The actuators block represents all devices that interact with the environment. Some examples in a home environment are light controllers, window shutters, ambient audio sound, door controllers, television, and others. Actuators can be classified in two categories: *system function actuators* and *system visibility actuators*.

The *system function actuator*s are the ones that perform a particular action in the environment. They need to send information of the execution of its actions as an event to the event database, called a *feedback event*. This *feedback event* allows the confirmation of the execution of this specific action. The *feedback event* also permits the monitoring of the system functionality, allowing the assessment and backup of system behaviour. The detection of malfunctions can be done in case of irregular activities when compared with the normal functionality.

The *system visibility actuators* can execute functions as well as let the user know what the system intends or is really doing, i.e., they show the internal state of the system in a comprehensive way to users. This capability enables building up the trust of the user in the system as explained by Antifakos [Antifakos, 2005]. Some examples of these actuators are: active displays, OSD messages on the TV, information LEDs.

Actuators can receive instructions from the upper layer (action coordinator) or autonomously decide to act, by analysing the sensor event activities. The process of acting without the upper layer involvement is called reflex action. This process mimes the process of reflex action that happens in the human body. In the reflex action, in the human body, the signals travel from the sensing area to the muscle by the spinal cord (reflex arc[1]) and without intervention of the brain [Purves, 2004].

These reflexive actions make the system more reliable, faster and more independent, i.e., they can do the basic and direct functions, behaving as if it was in a safe-mode operation. The reflex action may be configured by the upper layer that is responsible for embedding the default

[1] More information available on-line at: http://en.wikipedia.org/wiki/Reflex

behaviour of the system within the actuators. This behaviour is determined by the analysis of the event database.

3.3.3 Data and Hardware Management Block

The data and hardware management block implements the local view perspective autonomic functionality. This block coordinates a collection of sensors and actuators, managing the event database and coordinating the device actions. In practice, these functions can be located on a router/gateway in a TCP/IP environment, or in a network Zigbee coordinator or in a network protocol bridge (*communicator devices*).

This block is responsible for providing the network settings, such as: the network address, binding devices. It is also responsible for pooling the data of the *environment variable sensors* and setting the definitions of the alarm generation schema to create events. As this block manages the event database, coordinates the actions to be taken and communicates with the context services, it allows the optimization of the local system by defining default actions in the actuators, taking into account the goals of the context services. The self-protection feature of a sub-system is also applied in this block, i.e., it defines which devices can send/receive information and which context services are allowed to use this data and hardware management block.

Finally, this block can be used as a context-provider to the upper layer context services [Huebscher et al., 2005], but it can work without the connection to the upper layers, managing the hardware autonomic functions and, therefore, making the system upper layer less dependent and more robust.

3.3.4 Context Services Block

The context service block is responsible for creating the important context-awareness of a ubiquitous system. It can select the best context provider, as described by Huebscher

[Huebscher et al., 2005]. This block implements the global autonomic system with high level goals. It also allows the message associated with an action to be sent to the lower layer. Several context services can exist in ubiquitous systems depending on the type of application required. An example of a context service can be the detection of the presence of someone at home, i.e., it is possible to detect if there are some events recorded by the devices that were activated by any inhabitant at home. This information can be obtained from an alarm system context provider or can be obtained from the home automation context provider.

The context services are setup using high level goals. These services continue to monitor events and build a comprehensive interpretation of the analyzed information towards the creation of context awareness. Each context service can be viewed as a software component of the system. Some context services can make other context services obsolete, by including the same service, but showing better results.

The Context Service is responsible for the processing of raw events supplied by the *Data&Hardware Management Block*, by filtering events and supplying the context via an Application Programming Interface (API) service to client application.

3.3.5 Applications Block

The applications block take advantage of the context services. For the reason that environment data has already been processed, the application is only focused on the desired functions, resulting in simple applications that can run even in the smallest devices, i.e., applications that use very few resources. For example: an application to shutdown all lights and close all window shutters in a house becomes very simple if a context service is available informing that nobody is at home at a given time. This however, assumes that the communication capabilities are always available on in order to allow the application to reach the context provider. Moreover, an application can use, at the same time, several context services to improve and expand its functionality.

3.4 System Design Characteristics

To implement an autonomic ubiquitous system, some characteristics are required in order to achieve a robust system. Some of those characteristics are:

- Distributed – The system should work in a distributed manner, avoiding centralized processing units dependence. When this cannot be avoided, the system should guarantee that the more basic functions are implemented in a distributed way, guaranteeing, therefore, its basic, low level, functionality.

- Impromptu Interoperability – The system should allow, via interfaces and data availability, to be included in a more complex structure that can augment its functionality.

- Scalable – The system design should maintain the same structure in a small scale and in a large scale. This characteristic is achieved by combining the interoperability and distribution of the system, because the first guarantees that the system can be used in combination with other systems and the second characteristic distributes the data processing load, reducing the needed processing power of a high level controller unit.

- Fault-tolerant – The system needs to be fault-tolerant by allowing the system to degrade gracefully, i.e., losing a component cannot produce a catastrophic global system failure. This characteristic defines that the system should continue working with reduced functions, ignoring the failure of a component. This characteristic is different from the self-healing which tries to correct the problem.

3.5 Related Middleware Approaches

Some middleware approaches that are emerging are focused on some of the characteristics mentioned before. These approaches try to provide an abstract middleware layer that allows a smooth interaction between underlying hardware and the application software. Some of well-known middleware protocols in this field are: LonWorks, X10, UPnP, OSGi, Jini and oBIX. Some of these are oriented to control and automation functions, while others concentrate on data streaming and media management.

The LonWorks[1] is a networking platform specifically created to address the unique performance, reliability, installation and maintenance needs of control applications. The platform is built on a protocol created by Echelon Corporation[2] for networking devices over media such as twisted pair, powerlines, fiber optics and RF. Essential elements in this middleware system include LonWorks protocol; the implementation of the protocol (usually packed in one single chip called Neuron Chip); and the network components. The LonWorks protocol[3] is used in the communication among LonWorks devices. The design of the protocol follows Open Systems Interconnection (OSI) model. The protocol provides two sets of communication services: the first one enables application programs in one device to communicate with other devices over the network, the second one supports network management services, including addressing reconfiguration and controlling device application programs. LonWorks is an important solution for control networks which was developed for decreasing the cost and design complexity of distributed control systems. The protocol itself is media-independent, allowing LonWorks devices to communicate over any physical transport media.

The X10[4] is an international and open industry standard that also focuses on the control of the devices. It primarily uses power line wiring for signalling and control, where the signals involve brief radio frequency bursts representing digital information. A wireless radio based protocol transport is also defined. The commands and signals are broadcast throughout the medium and picked by the actuators, i.e., a single message can be used by one or more devices, but the failure

[1] LonWorks Middleware documentation available online at: http://www.echelon.com/support/documentation/

[2] More information available on-line at: http://www.echelon.com

[3] LonWorks protocol is also known as LonTalk protocol and ANSI/EIA 709.1 Control Networking Standard.

[4] X10 Theory. Available on-line at: http://www.smarthomeusa.com/info/x10theory/

of a single receiving device does not affect other receiving devices. This characteristic is very interesting, but the lack of the actuators feedback limits the needed robustness. Despite the severe limitations of this standard protocol, it is broadly used due to its simplicity and low relative cost.

The Universal Plug and Play (UPnP[1]) is an open-standard technology designed to enable the simultaneous control of multiple networked home appliances. It is a distributed, open networking architecture that uses the TCP/IP and the WEB to enable seamless networking and, in addition, to control and data transfer among networked devices in the human environment spaces. The UPnP architecture supports zero-configuration and automatic discovery by:

- Dynamically joining a network;
- Obtaining an IP address using the network DHCP server or the AUTOIP method;
- Announcing its name and transmitting its capabilities upon request;
- Learning about the presence and capabilities of other devices.

The UPnP protocol uses the Hypertext Transfer Protocol (HTTP) and the information has an Extensible Markup Language (XML) syntax.

The main disadvantages of the UPnP is the need of the devices to have an IP network interface, which increases the cost of the devices, as well as the lack of authentication/security. The IP network interface cost is more important in small, simple and cheap devices where the inclusion of an internet connection substantially increases the final product cost. To reduce this limitation, the UPnP allows the usage of HTTP over UDP (HTTPU[2]) that allocates a smaller memory footprint and, therefore, reducing the UPnP usage impact in the final device price. The lack of authentication/security is a constraining issue, because the UPnP is supposed to be used in a non-dedicated network where other devices share the medium and therefore making it prone to attacks. As a result, some UPnP devices are shipped with UPnP turned off by default, as a security measure.

---

[1] More information available online at: http://www.upnp.org/

[2] HTTPU is an extension of the HTTP/1.1 protocol using UDP as the data transport instead of the usual TCP protocol. HTTPMU is a variant of HTTPU that uses IP multicast. HTTPU and HTTPMU are not standardized and are specified only in an Internet-Draft that expired in 2001.

The Open Services Gateway initiative (OSGi[1]) or OSGi Alliance is an open standards organization founded in March 1999. The OSGi specification provides an open-standard for remote programmable devices, using JAVA and a meta-data file to describe services. It facilitates the installation and operation of multiple services on a single service gateway. The gateway can be any of capable devices from set-top-box or DSL modem or PC to a dedicated home gateway. The gateway platform execution comprises the downloading of software, application life cycle management, security of the programming environment, management of device drivers for external devices, configuration management, user management and a remote administration.

The OSGi framework consists in two types of entities: services and bundles. Services, in OSGi, are actually Java classes that perform certain operations. A Bundle, in .JAR file, contains resources implementing zero or more services and a Manifest file for description, so that the framework can correctly install and activate the bundle. They are published in a provider site where they can be downloaded to a typical gateway to provide the desired services.

The OSGi specification provides a very good and general framework for the integration of heterogeneous devices as described by Lee [Lee et al., 2003]. The main disadvantage is related with the centralized operation in a single gateway and the related robustness problems, but the centralized design allows the OSGi framework to implement gateways among other different middleware protocols.

The Jini[2] is also a JAVA base middleware, but it diverges from OSGi, because it is a distributed platform. Jini adds to Java and JavaRMI[3] other services which are useful for complex distributed applications (naming, lookup and discovery services, eventing paradigm, distributed garbage collection and transaction scheme).

---

[1] More information available online at: http://www.osgi.org/

[2] More Information available online at: http://www.sun.com/software/jini/specs/

[3] The Java Remote Method Invocation (JavaRMI), is a Java application programming interface for performing the object equivalent of remote procedure calls, that allows a program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction.

An environment needs to have three main JINI components: Services, Clients and Lookup Services (LUS):

- A Jini service is an element playing as a front-end of network resource, producing service descriptions and representing services. They need to be registered, at least, on a network Lookup Service to be available for usage by the clients;
- A Jini client uses the services. Firstly, a client searches for a given service in one or more Lookup Services. When a positive matching occurs, the client downloads a proxy of the service from the Lookup Service and executes it in client's JVM with the help of the JavaRMI mechanism;
- A Jini Lookup Service is directory or index of available services where service description, via a proxy application, can be downloaded.

The JINI distributed platform depends on the LUS to allow the clients to discover the services. In order to increase the robustness in JINI distributed environment, it should have more than a single LUS available. The JINI java implementation requires that the devices have a Java Virtual Machine (JVM), which limits the usage of very low resource microcontroller based devices.

The Open Building Information Exchange (oBIX[1]) uses the XML and Web Services to allow the exchange of information among intelligent buildings, enable enterprise application integration and achieve true systems integration. The oBIX provides a publicly available web services interface specification that can be used to obtain data in a simple and secure manner from access control, utilities and other building automation systems. In addition, it provides data exchange among facility systems and enterprise applications.

The oBIX 1st release provides a normalized representation for the three common elements to control systems:

[1] More information available online at: http://www.obix.org/

- Points: representing a single scalar value and its status – typically these map sensors, actuators or configuration variables like a setpoint;
- Alarming: modelling, routing and acknowledgment of alarms. Alarms indicate a condition which requires notification of either a user or another application;
- Histories: modelling and querying of time sampled point data. Typically edge devices collect a time stamped history of point values which can be fed into higher level applications for analysis.

The oBIX architecture consists in three main blocks or layers: Objects Model, Contracts and Bindings.

The Object Model layer defines primitive blocks to model the systems behaviour semantics.

A contract is a list of all patterns to which a complex piece of data conforms. Contracts are used to describe standardized structures such as points, historical trends and alarms; they are also used to describe proprietary vendor data. The main point of the contract is that new ones can be introduced without changing the oBIX schema. The bindings block consists in the HTTP/HTTPS[1] protocols and SOAP[2]/WSDL[3] that allow other devices to inter-connect between systems. It also facilitates the system integrator applications, e.g., the deployment of a home automation system.

The role of the web service in oBIX is to guarantee the interoperability of home appliances. Physical components such as sensors and actuators are not required to understand Web Services, but their controllers are, as well as the new devices entering the home network. oBIX gateways also need to understand web services in order to communicate with the outside world and coordinate external services.

---

[1] Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) is a URI scheme used to indicate a secure communication such as payment transactions and corporate information systems.

[2] Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

[3] Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services.

3.6 Summary and Conclusions

In this chapter the autonomic computing system characteristics were described in order to create a real ubiquitous computing system. The self-management allows the implementation of a genuine ubiquitous system.

An overview of the autonomic capability was presented. The main four autonomic capabilities: self-configuration, self-healing, self-optimization and self-protection were described in two different perspectives: global view perspective and local view perspective. These two perspectives present the same self-management objective, although at a different level. This division allows the creating of two levels: the implementation of the systems, i.e., a system can only implement the local view perspective autonomic capabilities and later, via integration, the global capabilities can be added. The global view perspective capabilities can also be implemented in systems that are not equipped with the local view perspective capabilities. In order to accomplish this, a gateway/bridge device needs to complement or emulate the system missing capabilities.

The system block diagram (Figure 3) shows the main components of the implementation of an autonomic ubiquitous system. The diagram shows how data flows from the sensor to the applications, via context services and also how the actuators receive information from the sensors. In the system block diagram it is fundamental to observe two important features: the reflexive action and the actuators feedback event. The reflexive action allows sensors to generate actions in the actuators without intervention of upper-layer or high-level components of the system, thus dramatically improving the robustness of the system. The actuators feedback events feature is also important as it allows the verification of the correct operation of the system, by monitoring and predicting its behaviour.

Moreover, the design characteristics of an autonomic ubiquitous system were enumerated. On one hand, the characteristics of a distributed, impromptu interoperability, scalable and fault-tolerant system are essential to build a robust design. On the other hand, these characteristics increase the complexity and the cost of such solution.

Some available middleware protocols were analysed, as they present one or more characteristics in the design or in the implementation that can be used in an autonomic ubiquitous system. The middleware protocols examined range from the simple and relatively old protocols, like X10, to more sophisticated and recent ones, like Jini or oBIX. Several other middleware systems are also available and were not mentioned. The presented middleware are the most common and widely available.

The design characteristics to create an autonomic ubiquitous system and the middleware evaluation show some common features, despite the fact that the middleware protocols were not created taking into account an autonomic behaviour. The examination shows that, in some cases, by adding some bridge device, it is possible to include extra functionality that allows the systems to be autonomic. The solution for the usage of smart bridges/Gateways is, in fact, utilized by the OSGi framework and the oBIX. The usage of gateways is a solution for the integration of other sub-systems, but it is important that these sub-systems provide the required low-level characteristics to allow the autonomic ubiquitous system implementation.

The next chapter is focused on the hardware that enables the vision of a ubiquitous philosophy, which are the building blocks of ubiquitous systems.

4 Ubiquitous System Building Blocks

In this chapter, the designed hardware that enables the vision of ubiquitous philosophy is presented. The details of how sensors can be inserted in the environment are presented, as well as how data can be captured and transmitted. Sensor and actuators functionality are assessed and classified. Communicators are presented as an helper in the construction of the system.

4.1 Interfacing with the environment

The sensors and actuators play an important role in the implementation of an autonomous system that interacts with the environment. These devices limit and define, in some degree, what the system perceives and where it can act upon. Sensors and actuators are an important part of any system and they are widely used in robotics, automation and engineering.

Acting and reacting to data supplied by sensors is one of the basic properties of almost all intelligent systems. In the natural environment, the capability of adapting to new situations is an important advantage and two different strategies can be observed: some organisms use a bigger and more complex brain (data processing unit) which is ideal to adapt to rapidly changing situations; other organisms improve, by an evolutionary process, their sensors and actuators to get a better performance or even to get new data from other types of sensors. This strategy is limited to a slow adaptation rate (limited by the rate of new generations). However, it presents a simpler, more energetically-efficient and more robust solution.

The use of more sensors and higher diversity allow data processing units to be simpler, i.e., more accurate information can be acquired from the environment; however, the amount of data to be processed is larger.

4.2 Sensing and acting in a ubiquitous system

Sensing and acting in a ubiquitous system involve some technological, social and economic requirements to be taken into account, as presented in Table 1. These constrains limit and define the quality of the context that can be acquired and achieved. Some of the requirements are more important for mobile devices, others are more relevant for stationary devices when included in a ubiquitous system, e.g., the power consumption is a requirement that is important for mobile devices but it is also significant when they are applied in a large quantity, as stationary devices in an ubiquitous system environment. These requirements must be examined and studied as they relate to each another. A balance among them must be taken into account because their implementation can be impracticable in the same device.

Table 2 – Requirements when designing and building sensors and actuators

Requirements when building sensors and actuators
• Design and Usability • Robustness and Reliability • Unobtrusiveness • Social Acceptance and user concern • Accuracy and Precision • Openness and impromptu interoperability • Price and Development Cost • Energy Consumption • Start-up Time and Calibration • Portability, Size and Weight

4.2.1 Design and Usability

When designing a device, the inclusion of a sensor can limit its usability. For example, in some mobile phones with automatic screen rotation, when user flips 90° the device, the screen automatically flips to wide screen format. This functionality assumes that the device is being manipulated in a stand normal user position and that the device is facing the user. The problems start when the device is being manipulated by a user that is lying on the couch. In this position, the automatic functionality simply does not work well and the problem ends when it is deactivated by the user.

The inclusion of sensing technology should be placed where the handling does not interfere with measurements, e.g., regarding a device that measures the light or captures sound, the sensor windows should not be placed in a way where they can be obfuscated by the hands of the user.

The inclusion of any sensing technology should not affect the usability of a device and the added value should be greater than the constraints that are caused by that technology.

4.2.2 Robustness and Reliability

Devices should be robust to be useful in everyday environments. The robustness of a device should be considered in its capability to suffer some unintentional abuses or bad operation (e.g., in a normal daily situation, where it is dropped, shaken, heated, etc) and still keep its normal operation. Reliability is the characteristic of a system to perform and maintain its functions in routine circumstances, as well as unexpected circumstances.

Nevertheless, the robustness is not only defined by its mechanical strengths or electrical appropriateness, but also by the quality of the acquired data. Data processing needs to be prepared to work with unreliable data and to give some indication of the quality of the captured data.

4.2.3 Unobtrusiveness

Building sensing into devices and environments unobtrusively is a goal that is present in many Ubiquitous Computing projects and is also related to the concepts of calm technology as introduced in [Weiser and Brown, 1996] and in ambient interfaces [Wisnesk et al., 1998], [Gellersen et al., 1999].

The unobtrusiveness is a significant aspect because the main objective in the creation of a ubiquitous system is to free the attention of the users from the technological details. Devices that require a special attention from users need to alert them in a soft manner. This signalling must be appropriate to the environment where it is included and must be proportional to the level of urgency.

4.2.4 Social acceptance and the concern of the user

The determination of the position of the user, location or capture of audio or video in some circumstances is not socially acceptable. When sensors capture personal information of the users, there is a need to let them know that that information is being acquired. The same concern is applied into actuators that should not expose personal information in a public way.

The devices need to inform the user that data is captured taking into account the unobtrusive aspect. The social acceptance should always be considered and the right balance needs to be achieved between functionality and the concern of the users for a particular environment and application.

4.2.5 Accuracy and Precision

Traditional sensing systems require a specific accuracy and precision of sensors, and that is determined according to each particular application. In a specific system where the requirements in

accuracy and precision are fixed the selection of sensors and actuators is simplified. But in a Ubiquitous Computing environment, sensors and actuators can support different applications, as long as their accuracy and precision is sufficient. To help the autonomic decision in selection of sensor sources, sensors must also supply its accuracy and precision. Having this information about the data, the system can adapt to the sources that have the most appropriate data quality.

4.2.6 Openness and impromptu interoperability

The openness to future integration of devices is a very important aspect in the development of a ubiquitous environment because the requirements and conditions, during the development of sensors and actuators, for this type of systems cannot be clearly determined upfront.

The devices need to transmit the information in a standard protocol when possible. When that situation is not feasible for technical or economic reasons, the protocol details should be open. Sometimes, for commercial reasons, the complete details are not available. In this situation, a Software Development Kit (SDK) should be available for developers to integrate these devices in their application.

As important as the data protocol or SDK availability is the way data is supplied by devices. The devices should emit data when it is generated, as opposed as to a request/reply method. If devices supply information in broadcast mode, this can be captured by future devices that may be included in the system at a later stage. This greatly simplifies interoperability and allows monitoring of the functionalities which help the autonomic implementation.

4.2.7 Price and Development Cost

The Price and Development Cost can shape and obstruct the inclusion of some options in the adopted devices. The implementation of a distributed network of devices has a greater

development cost in comparison to a development of a master/slave network. Generally, the creation of more complex devices results in a greater cost of development.

The price can also influence the technology used in devices. A more sophisticated protocol needs more processing power and consequently a greater price in processing power. The price, in some degree, limits the usage of a single and universal communication protocol by these devices.

Adding extra resources that increase the final price of devices always depend on the benefit that an application gains by being context-aware. The additional value is highly subjective because it is dependent on the significance that people perceive from the additional functionality enabled through context. Moreover, many variables affect each other so it is more of a design decision than a calculation.

4.2.8 Energy Consumption

As described later, power consumption is a requirement that is important for mobile devices but it is also important when they are applied in a large quantity, as in the case of fixed devices in a ubiquitous system environment.

Lower energy consumption limits the type of the processor unit to be used or the clock speed that it can operate, which leads to the usage of simpler algorithms. The sensors selection is also affected by power restriction.

Devices should allow power control over modules that are not always in use. This let power consumption to be shaped to the needs of a particular application. Overall, the ubiquitous devices have to be developed to minimize power consumption.

4.2.9 Start-up Time and Calibration

As devices increase in complexity the Start-up Time increases proportionally and this latency can have a significant negative impact on overall system performance and functionality. Some sensors can involve complex electronic, mechanical or chemical processes to measure certain environmental conditions which can also deteriorate the system readiness. This delay is normally caused by the auto-calibration procedure required and implemented in the sensors.

The usage of sensors that are calibrated during production can minimize this problem but increases the final device price. In some situations, the calibration can drift over time, requiring regular calibrations. To minimize calibration problems, the system should use relative readings or variation in readings, i.e., derivative readings over time, in opposition to rely on absolute values.

A long start-up time delay can make the integration more complex as the system needs to take into account the state devices. In order to overcome this constraint associated with the start-up state, the device should supply neutral readings during the boot-up time or replicate readings captured during the last usage. In addition, the quality indicator of the supplied readings during the boot-up should be low, allowing the system to select other sources of data until the device complete its start-up or calibration.

4.2.10 Portability, Size and Weight

For mobile devices, the usability is directly related with the portability which can be defined by the size and weight.

Size and weight are not related with the sensors but they are indirectly to the perception algorithms used. More complex and computing intense algorithms need more processing power or storage. More resources lead to greater demand on the processor that ultimately leads to higher power consumption and, therefore, to a heavier and bigger device, due to the batteries required.

In some circumstances, it is possible to reduce the mobile device resources by enabling them to transmit the captured data to a remote processing unit where the processing limitation is not an issue. This strategy not only solves in some degree the mobile limitation, but it allows the implementation of other functionalities by using the captured data in other contexts.

4.3 Technologies to sense the environment

The most relevant sensing technologies are widely used in robotics, automation and process control. In autonomous robotics, the sensors supply the essential needed information for proper decision making support. In ubiquitous systems the sensors also provide the information that helps the system to determine the context of a given situation.

Table 3 – Sensor technologies that can supply data to a ubiquitous system

Sensing Technologies
• Light and Vision • Audio and Noise • Movement and Acceleration • Position and Location • Magnetic Field and Orientation • Touch and User Interaction • Temperature and Humidity • Weight • Motion Detection

Over time, sensing technology has presented significant improvements with respect of physical size, weight, power consumption, processing requirements, interfacing options, reliability,

robustness and price. These improvements facilitate the inclusion and deployment of these technologies in ubiquitous systems.

In Table 3, some sensing technologies are listed whose aims are sensing the surrounding environment characteristics and detecting situations and actions from the interaction of the users.

4.3.1 Light and Vision

Light measurements are relatively simple to obtain. The capture is not limited to visible light and it can range from Infra-Red (IR) to Ultraviolet (UV) wavelength. This wide wavelength range allows the system to capture information that is not noticeable to human eye.

The light optical sensors can be: photo-diode, colour sensors, IR sensors, UV-sensors, among others. These sensors can characterise light in: intensity, density, wavelength (colour temperature), light source (sunlight, artificial light).

Vision sensing is essentially a matrix of light sensors, although not technically implemented that way. Almost all sensing made using light sensors can be emulated using a vision sensing method as described by Lima [Lima et al., 2000]. The camera modules are getting less expensive and becoming almost ubiquitous in mobile phones, but their great potential is not yet fully explored. They have higher demands on processing and storage that cannot be met by low-end microcontrollers which limit its use in some applications.

Video Camera monitoring makes users unconformable which also limits its use in human environments. In these situations, the light sensors have greater acceptance by the end-users.

4.3.2 Audio and Noise

Sound is another source of information that can help in the determination of some context situations. For audio capture, it is used a microphone, a preamplifier (preamp), an automatic gain control (AGC) and a sample & hold circuit that allows the digitalization of the signal. Simple processing can be directly implemented in analogue circuits but normally all the processing is done in a microcontroller.

The features that can be extracted from the sound can go from volume (noise) and spectrum to full audio processing including speech recognition. The processing power required can be from very low to extremely intensive, depending on what type of information is wanted to be extracted from data.

The combination of several sources of sound can provide information about the location of an audio source as described by [Svaizer et al., 1997].

4.3.3 Movement and Acceleration

Movement and Acceleration can be very helpful sources of context information. The sensing of motion and acceleration can be captured by a wearable device or using a vision sensor. Sensing these movements and accelerations in wearable devices can be implemented using different sensors, such as: motion switches, accelerometers or gyroscopes.

Motion switches offer a simple form for detecting movement. They are attached to a device in such a way that the anticipated movement, resulting in changing the position of the conductive element in the switch. This then leads to a change in the state of the switch (e.g. from open to closed or vice-versa). These sensors can be used to wake-up microcontrollers from energy saving sleep mode.

Accelerometers are available as integrated micro-machined devices combined with driving electronics. They can measure acceleration in one, two or three directions, shifted 90°. This class of sensor is very precise and in case of being used a three direction sensor, it is possible to determine the orientation of the movements.

Gyroscopes are another option when angular velocity is of interest. These devices are very precise, but generally more expensive and need more power. They usually supply an analogue signal that represents the angular velocity in volt per degree per second.

4.3.4 Position and Location

Position and location is one of most important features of the context construction. A lot of ubiquitous system applications depend on this information to perform well.

For sensing an outdoor location, the GPS[1] or DGPS[2] are the traditional technology systems used. Normally these devices require a relatively long start-up time (typically more that 30 seconds) and have considerable power consumption. Depending on the number of satellites which are visible, the accuracy is within a few meters for the GPS and some centimetres for DGPS. Another way of determining the position or location is to use the GSM signal strength and the GSM Cell ID which are able to determine an area where it is located even if it is inside a building. In outdoor systems the accuracy of the position can range from town level down to centimetre level.

Several technologies are available for indoors the location and position system using beacons of IR, Ultra-sonic, Wi-Fi AP or Bluetooth, among others. They use the signal strength and orientation to allow triangulation to determine position. The aim of most location systems is not to supply with

[1] Global Positioning System (GPS)

[2] Differential Global Positioning System (DGPS) is an enhancement to Global Positioning System that uses a network of fixed, ground-based reference stations to broadcast the difference between the positions indicated by the satellite systems and the known fixed positions. These stations broadcast the difference between the measured satellite pseudoranges and actual (internally computed) pseudoranges. The receiver stations may correct their pseudoranges by the same amount.

an absolute location coordinate but geometric location information that can be used as an index to access further information.

4.3.5 Magnetic Field and Orientation

Different types of sensors are available to detect magnetic fields. Some are designed to detect the earth's magnetic field whereas others are constructed to detect the proximity or change of a generated magnetic field. Hall-sensors detect the flux of the magnetic field applied. Sensors that detect the earth's magnetic field are the basic building blocks for an electronic compass. The output is related to the direction of the magnetic field and can be used to figure out north direction.

Sensors that use the magnetic field of the earth may fail in places where there is a lot of electromagnetic interference and that should be taken into account. Hall-effect sensors, on the order hand, are very robust and are widely used in the heavy industry.

4.3.6 Touch and User Interaction

The Touch and User Interaction is one of the most important ways to acquire the context of a situation. This is not related with the technology that is used to capture the user interaction, but it is related with the capability to capture and report interactions that are the result of a normal operation with a specific device.

Actions like opening a door, switching a light, turning on the TV, among others have a very high importance in context building. The analysis of the way users interacts with common devices can provide relevant information and therefore it is an important data source.

4.3.7 Temperature and Humidity

Temperature can be sensed using very simple thermal resistors or more sophisticated temperature sensors with built-in driving circuits. Such temperature sensors are available with analogue or digital interfaces and offer high accuracy.

Sensors for humidity are more complex and also more expensive than sensors for temperature. Humidity sensors are available based on capacitive and resistive technologies and also as modules that provide an analogue or digital output that is proportional to the humidity level. Humidity sensors react in a matter of seconds to changes.

These environment measurements can help to determine if a space is being used by any human as the humidity and temperature tend to increase in that situation.

4.3.8 Weight

The weight measure is obtained using load cells. These convert a force into an electrical signal. These types of sensors are widely deployed in industrial systems. Load cells can measure from milligrams to the weight of several hundred tons. The resolution is dependent on the range and also on the quality of the device. Load cells are available based on different technologies and also using different interfaces. Commonly used industrial load cells, based on resistive technologies, incorporate a Wheatstone-Bridge and provide an analog output signal. Off-the-shelf load cells are available in different physical shapes and sizes.

The weight in combination with other measurements can be used to associate actions to any particular user, contributing to the enrichment of the context.

4.3.9 Motion Detection

Detecting the presence of a user in a space is very interesting to many applications in Ubiquitous Computing. Motion detection of subjects and objects in a certain space can be assisted using different technology.

One way to detect motion is the use Passive Infrared Sensors (PIR). These sensors detect changes in the heat flow in the environment and can therefore detect humans and animals moving in the detecting region of the sensor. They provide digital output offering the binary information whether or not someone entered or left the detecting region. These sensors always have a main sensing direction and are available with different lenses offering observation angles of 30° to 180° and ranges from 2 to 15 meters.

Motion detection can also be implemented with vision systems but the PIR sensor presents simpler and more affordable solution to users.

4.4 Executing actions in the environment

In ubiquitous systems, interactions with users are fundamental. These interactions are the final results provided by the system to users, i.e., only a system that interacts with its environment can have some effect on it.

Any device that can change an environment variable, which may be observed or felt by a user, can be used as a potential actuator. There are also actuators that can interact with the environment in a way that is not detected by humans. These are normally used as a support to other sensors or actuators. Some examples of this type of devices are some ultrasonic or infrared beacons that supply information to wearable tags, allowing the identification of a person, a place or a position coordinates.

The actuators that may have more impact in users are the ones directly related with the importance of some human senses. Humans perceive the world through the following senses:

- Sight
- Hearing
- Smell
- Touch
- Taste

Although other human senses also exist, they are related with internal human body sense, e.g., balance/acceleration, pain, kinaesthetic sense and others.

The most useful senses for a ubiquitous system application are those that can reach the users effortlessly, i.e., without the need of a very specific action from the user. This way, the vision (sight) and sound (hearing) are the main ways to communicate with a human. The smell can also be useful but tend to have a very slow change rate and propagation velocity. The use of touch can be exploited by wearable devices by alerting user to a specific event using vibration. Taste requires a specific action from the users and it is a sense not normally used to acquire information related with environment.

Related with the human main senses, almost all devices include some type of interface that whether creates some visual representation or produces some type of sound. The interfaces are normally used to expose specific information related with device functions, but many times they have much more display potential.

Actuators need to be constructed to support visualizations of information or emission of sound in an open way, i.e., the devices that have to make available their actuators to other systems. The development of these extra features does not require much effort.

4.5 Embedded environment sensors and actuators

The ubiquitous system environments are filled with sensors and actuators. Hiding sensors in the environment is in most of the situations possible. They can be placed behind mirrors, with very small windows (pin holes) or concealed in other types of devices. However, hiding sensor and actuators in a way that users can not detect it is not essential.

Figure 4 – Some examples of commercial sensor and actuator devices. a) Infrared light barrier; b) Dom camera; PIR sensor with hidden camera; d) Light Switch; e) Door Magnetic Sensor; f) Door lever with switch contact; g) floor switch; h) sound colon with hidden camera; i) light; j) television; k) lava lamp

The essential characteristic that must be addressed is the inconspicuousness of the interactions between the system and the users. The inconspicuousness means that user does not need to think about the action to allow it to be executed in a natural way. The user activities are read by the system and it responds using unobtrusive means.

In Figure 4 there are presented some examples of sensors and actuators commercially available that can be used to capture user actions. Some of these devices are used as a part of security systems and others are used in home automation.

Devices such as the infrared barrier, cameras, PIR or floor switches can be placed strategically to capture user movements or positions. They can easily determine the presence of a person in a room.

Another way to effortlessly acquire user actions is to acquire actions that are already part of every day activities, e.g., the switch of a light or the opening of a door. These actions are so common on a daily basis that a user executes them without thinking. Devices like the door lever with an electric switch or a magnetic door sensor are examples of devices that can capture that type of actions.

The users can receive information from already existent devices in human environments. Almost everywhere there are displays or audio sources that can be used to report to users. This interaction can be very direct, e.g., with a specific text or audio, or can be done in a more indirect way, e.g., the flickering of a light or a change in colour of a lava lamp (Figure 4 K).

4.6 Challenges when building a system

The building of a ubiquitous computing system using devices from different suppliers and technologies presents some challenges. These systems need knowledge from different fields; these can go from the electronics, networks, computer science, and artificial intelligence to others. Several problems need to be solved before the ubiquitous computing vision can become true, e.g., the

efficiency problem in managing a big number of devices from different manufacturers without standardization.

Table 4 – Table comparing 2.4GHz Wireless Networks

2.4 GHz wireless network	Wi-Fi	Bluetooth	ZigBee
IEEE standard	802.11 b	802.15.1	802.15.4
Max. Transfer rate	11 Mb/s	723.2 Kb/s [1]	250 Kb/s
Ave. range (m)	100	10-20	10-75
Max. data payload (bytes)	2304 (1500)	339 (DH5)	118
Overhead (bytes)	57	158/8	15
Min. data packet size (bytes)	49	126/8	15
Basic cell	BSS	Piconet	Star
Extension of the basic cell	ESS	Scatternet	Cluster tree
Max. number of cell nodes	2007	8	65536
Complexity	Very Complex	Complex	Simple
Protocol stack size (bytes)	> 250 K	≈ 250 K	28 K
Number of RF channels	3	79	16
Channel bandwidth (MHz)	25	1	2
Modulation	CCK	FHSS	DSSS
Channel access method	CSMA/CA	Polling	CSMA/CA
Approx. power consumption (mW)	30-100	1-10	I

In an ideal world, all these devices have the same protocol and all understand each other, but in the real world it is not feasible the idea of having a unique protocol that links all kinds of device classes, because every sub-system has its own objective. For example, in short-range RF standard protocols, wi-fi, bluetooth and zigbee, each one has its specific characteristics and applications as shown in Table 4 [Machado et al., 2007]. But the differences can exist at a physical level, e.g., connecting a wire device to wireless devices.

[1] Bluetooth V1.2 Max transfer rate using DH5 type packets with the highest data rate among all ACL packet types.

Devices that make bridges between protocols, named communicators, guarantee that communication channels are available among different devices with different technologies.

4.7 Summary and Conclusions

In this chapter it was presented the hardware characteristics and requirements taking into account the construction of ubiquitous system environments. The sensor and actuator characteristics limit, in some degree, what the system can perform.

The requirements that must be examined when creating the sensors and actuators were described. Not all the requirements can be concentrated into devices, but a balance between them should be achieved. The sensing technologies that can be used to supply the ubiquitous systems were also enumerated.

In this chapter it was also shown that the video and the audio are the principal means to inform users. Although audio is more intrusive than video, because in visual representation the human can choose not to look, that option is not possible to audio sources.

The need for communicator devices that can interlink devices was also introduced. This communicator allows the inclusion of management functions in already commercial available devices that were not built, taking into account the inclusion in a ubiquitous system.

In the next chapter, the management of devices is presented and it is shown a way to build systems in a low-level or local perspective. The autonomic management system is addressed on hardware level, i.e, the self-configuration, self-optimization, self-healing and self-protection built in them firmware of the board. The interfaces to data are also defined by showing how the following components can access to it.

5 Low Level Autonomic Management

In this chapter it is shown how the management system can be built in a low-level or local perspective. The autonomic management system is addressed at the hardware level, i.e., how the behaviour of the system can be embedded in the environment devices. This approach has implication on the firmware of the devices of the system and on the system network architecture capabilities, in order to achieve high robustness with graceful degradation of the whole system in case of failure of some components.

5.1 Low-Level Management System Principle

The Low-Level Management System implements the Local Perspective View of an Autonomic Ubiquitous System referred in a previous chapter. This management is focused on the data-flow, providing the upper-software layers with the means to implement the autonomic management.

The system characteristics to achieve a robust design start shaping the low-level system, i.e., the system needs to be distributed, impromptu interoperable[1], scalable and fault-tolerant. The Autonomic Nervous System (ANS) of a human body serve as an inspiration to functions that the low-level management system implements. The ANS works as the control system, maintaining homeostasis[2] in the body, i.e., it performs, without conscious control (automatic control) ensuring its stability, in response to fluctuations in the outside environment.

The ANS reflex arc is an example of an automatic action process, showing the steps that are involved in the execution of a reflexive action that must be present in the low-level management of an autonomic ubiquitous system.

---

[1] Interoperable is the ability of diverse systems and organizations to work together (inter-operate).

[2] Human homeostasis refers to the body's ability to refer physiologically its inner environment to ensure its stability in response to fluctuations in the outside environment and the weather. More information on-line in URL: http://en.wikipedia.org/wiki/Human_homeostasis

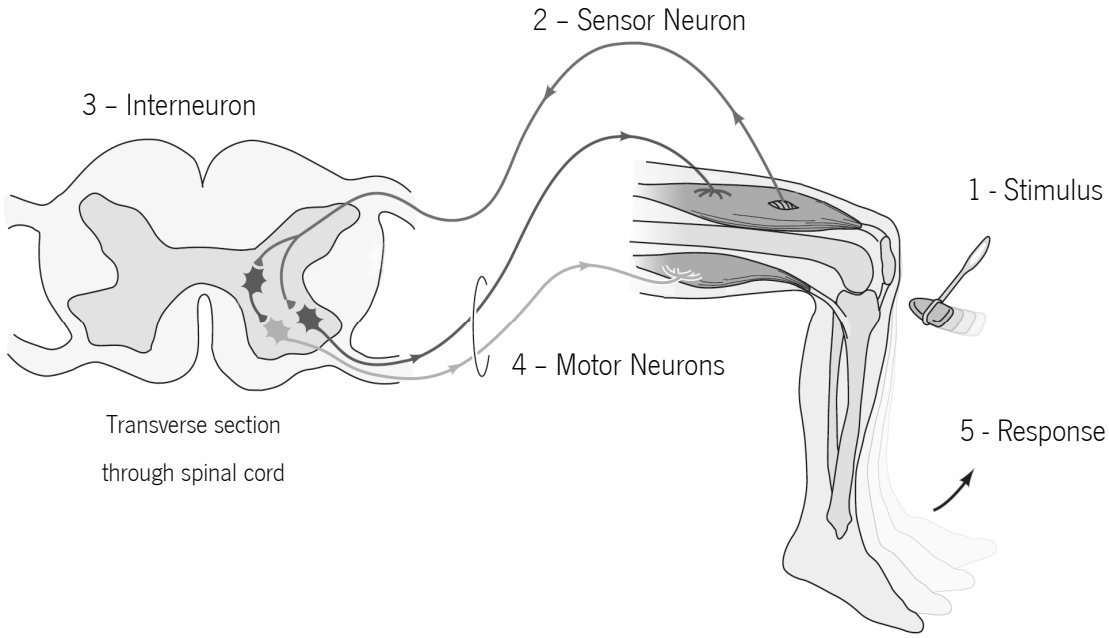


Figure 5 – Human Reflex Arc (knee-jerk reflex) [Purves, 2004]

In Figure 5 it is presented the Human Reflex Arc for the knee-jerk reflex. This is the simplest reflex arc which only uses two neurons in the spinal cord. The muscles in the leg have stretch receptors which react to a change in length of the muscle. When the hammer hits the tendon at the knee, it makes a muscle in the front of the thigh longer. That stimulates the stretch receptors in the muscle resulting in muscles in the front of the thigh contract and, at same time, the back muscle of the thigh relaxes. The muscle reactions make the foot jerks. The main function of this reflex is related with maintaining an upright posture, which requires fast and immediate responses to changes in the environment.

The stimulus travels through the sensor neuron to the spinal cord where it is processed by the interneuron and the result signals are transmitted to the muscles by the motor neurons that finally produce the muscle response. The information that arrives to the spinal cord is also transmitted to the brain, allowing the acknowledgement of the changes in the environment that may

result in extra action processed by the brain. These processes are explained in more details by Purves [Purves, 2004].

The Reflex Arc presents the characteristics needed as it senses the environment, transmits the information, makes local and simple data processing, executes actions and finally allows the remote or upper-layer control.

5.2 Hardware capabilities to imitate the ANS

The hardware needs to have capabilities that allow imitation of the ANS reflex arc process. The reflex arc process has; the sensing; the communication of the captured information; the local processing and the subsequent execution; and it reports and receives information from an upper-layer controller. The Table 5 presents the capabilities that need to be emulated by the hardware, and are explained as follows.

Table 5 – Hardware capabilities to emulate ANS Reflex Arc

Hardware capabilities
• Sensing and Actuation (using events) • Broadcasting Information • Configurable Binding between Events • Monitoring internal variables

5.2.1 Sensing and Actuating (using events)

Sensing information needs to be sent in bursts of data reflecting the changes in the environment parameter monitored or caused by an external stimulus, similar to the ANS Reflex Arc,

where that information is converted into electrical/chemical signals that are transmitted by the sensing neurons. These bursts of information can be interpreted as events that are sent immediately when changes occur, instead of requested by the processing unit. Because of this, the communication to sensing devices can be unidirectional (data going out of the sensing device), although bi-direction communication is needed to allow remote maintenance.

The actuation can also be event-driven. The actuation events are divided into two classes: the command events and the feedback events. The command events order the actuator to execute a specific action and these events are essential for the actuators functionality. The feedback events are the ones generated by the actuator after the execution of an action. In the ANS Reflex Arc of the

Figure 5, after the muscle action, this event is generated by the stretch sensor of the muscle. The same way, if there is a sensor available that measures the actions of an actuator, the feedback event produced by the actuator device can be optimal, resulting again in an unidirectional communication (only data is received by the actuator). The unidirectional communication allows the implementation of a lower power (mainly in wireless actuators because they do not need to transmit information), reducing the cost of the final devices, but limiting the configuration changes. Therefore, it should be only used when sensing and actuating is well defined and this way it does not need configuration adjustments.

The feedback events from the actuators are only generated if the actuator execution changes any parameter in the environment, similarly to the ANS Reflection Arc. This way, after a command event has been sent and a feedback event appears that indicates a change in the state of the actuator. However, when no feedback event is reported, it may indicate a communication problem or no changes needed in the actuator.

5.2.2 Broadcasting Information

The information in the human ANS Reflex Arc is processed not only in the Interneuron, but it is also transmitted to the brain. This indicates that the stimulus information is not only meant to reach a specific part of the body (spinal cord), but it can also influence all of it.

The same way, the information (events) generated in the sensing or in the actuation should be sent in broadcasting mode, i.e., the events are sent to all, immediately after their creation, without specifying any receptor.

The Broadcasting of information helps to solve the impromptu interoperability problem (devices that are ready to be used in new systems with new conditions) because the information goes out without destination and a possible new device can be developed taking into account the available data to integrate older devices.

5.2.3 Configurable Binding between Events

The binding between events imitates the signal processing of ANS Reflex Arc that occurs in the interneuron located in the spinal cord. This binding consists in the generation of one or more command events according to the type or nature of an incoming sensor or feedback event.

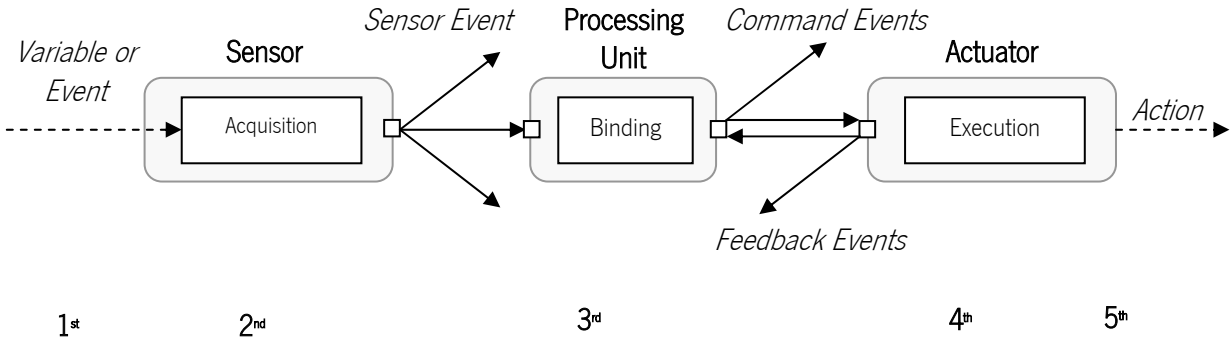


Figure 6 – Event binding in an external processing unit

In Figure 6 it is presented the flow of events, from the capture to the action, processed in an external processing unit included in a network device. The events can be processed by more that one processing unit because they are sent in broadcast mode. The number of steps needed to execute an action is equivalent to the number of steps in the ANS Reflex Arc.

In a system, the binding between events can be implemented in the sensor, in the actuator or in mixed function devices (devices that can sense and act in an environment). This characteristic allows the distribution of the processing load throughout the environment devices.

Figure 7, it is shown a device with the capability of sensing and actuate in the environment. This device executes internal binding, reporting at the same time the events, which in turn allows its external usage. If the network interface of the device has the capability of receiving external events, it can bind it with internal action (generating the appropriate command events).

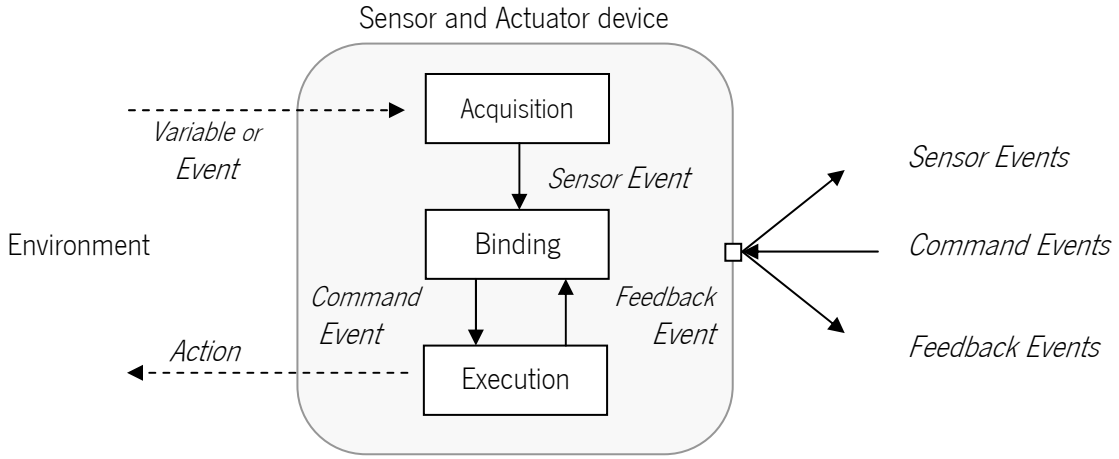


Figure 7 – Event bind inside a device

The binding between events and command events tend to be simple and limited by the computing resources of the devices that compose the system. They can be configured remotely by an upper-layer device that monitors the events and produces the bindings that can be configured in the devices. This allows the environment to embed the habits of the user. A basic binding is configured in the construction of the devices making them prepared to execute basic functions immediately from the first power-up.

5.2.4 Monitoring Internal Variables

The monitoring of internal variables consists in the capability of remotely access the device interval variables in order to acquire its internal state, analyse hardware resources and verify the log of possible errors. Some examples of internal variables are: inputs and outputs actual values; device temperature, supply voltage level and fuse state; firmware module versions, memory usage and stack level; and network metrics (received and transmitted bytes, error counts and buffer overflow counts).

The monitoring, in conjunction with the broadcasting data events, allows a controller device to implement the self-healing and the self-optimization of the system.

5.3 Network architecture capabilities

In order to implement the hardware capabilities with the event message mechanism, the network architecture must show some requirements to insure the correct functionality of the entire system. It is also important that the network architecture implements functionalities that allow the self-management of the communications. The network capabilities that help in the construction of a low-level autonomic system are shown in Table 6.

Table 6 – Network Capabilities

Network Capabilities
• Broadcasting without collisions • Distributed Network architecture • Automatic Address Assignment • Device remote interface blocking

5.3.1 Broadcasting without collisions

The broadcasting capability is essential to the implementation of the event methodology. When a device sends information to the medium, it must be assured that it reaches all the nearby devices in the network bus, because the event process philosophy suggested above does not allow the implementation of an acknowledged response.

The broadcasting of information does not need to be reliable, i.e., when an event is sent to the network it is not necessary to guarantee that all devices receive, because implementing the reliability requires more resources for every device.

Although the broadcast does not need to be reliable, it is important that the data packet sent by a device does not collide with other packets sent by other devices. When the network protocol used by the devices does not allow efficient collision detection, it may be possible to implement a medium allocation, achieving in such a way a collision less broadcasting.

5.3.2 Distributed Network architecture

A distributed network architecture is necessary to guarantee the robustness of the whole system. Protocols that need a server to manage a specific service must be avoided, because in case of server failure, the whole system stops working, i.e., with a centralized server the system has a single point of failure.

There is a tendency to implement centralized services. The main reasons are related with the simplicity, fast development time and their simpler maintenance.

In a centralized architecture the communication with the server has to be available all the time. In a distributed architecture, when a segment in the communication fails, the system is transformed into two independent working networks.

5.3.3 Automatic Address Assignment

The capability of a network node to assign its address automatically is essential to allow the device to be identified in the network. There are several protocols that allow this automatic address assignment. One of the most well-known is the AutoIP[1] in the TCP/IP protocol.

The AutoIP is a server-less method of choosing an IP address and it is used when no DHCP server is available. It selects an IP address and broadcasts it. Then it examines the in/out ARP packets to check if it is in conflict with the Link Local Address was selected. Then it protects or reselects the address accordingly. For outgoing packets, an ARP REPLY packet broadcast is sent onto the network (instead of uni-cast).

After an auto address, a management station can remotely change the device address to organise the devices.

5.3.4 Remote device network interface blocking

The remote device network interface blocking consists in the capability of remotely deactivating and activating the communications from and to a specific device. The capability can be used in two situations: to deactivate the communication from a device that is flooding the network with events and when there is a need to change for some time the communication protocol.

The flooding of events from a device can occur when a sensor is faulty and the processing filtering also fails. A monitoring station detects the problem and remotely blocks the network interface of the device until a firmware upgrade or service is carried out.

[1] AutoIP is an implementation of RFC 3927, "Dynamic Configuration of IPv4 Link-Local Addresses".

Some situations, such as the firmware upgrade of the device, may require a completely different communication protocol which is incompatible with the normal device communication. In this situation, the network interface blocking allows a temporary usage of the communication medium with different communication protocols. The interface blocking is necessary because of the very small resources available in the devices which limit the size of the communication stack protocols.

5.4 Teleswitch Device

In order to test the capabilities previously enumerated and validate the proposals of this project, a prototype device has been created. The prototype device objectives are the implementation of the capabilities of monitoring and acting in the environments, allowing users to interact with it in a natural and effortless way. Taking into account the various existing technologies in spaces inhabited by people, the electricity and its utilization in human environment (lighting spaces) was selected as the basis for the interaction with humans.

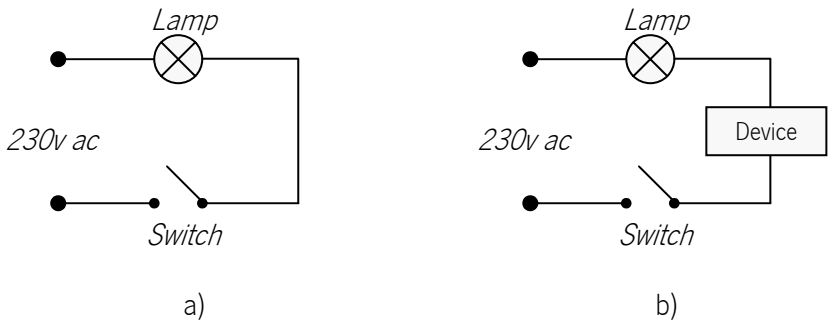


Figure 8 – Electrical switch/lamp circuit. a) Conventional circuit; b) circuit with a controller device;

The conventional circuit to turn a light on/off consists in a simple switch in series with the lamp and mains supply as presented in Figure 8 a). When the user wants to turn on the light, it toggles the switch and the electrical current flow, producing light in the lamp. This interaction, so common on people's daily routine, is executed several times a day without thinking.

The prototype device was placed in the middle of the switch/lamp circuit to capture the actions of the users, as shown in Figure 8 b), allowing later context processing and analysis. It also permits the remote control of the lamp state, interacting this way with the environment. The device was called Teleswitch, demonstrating the basic functionality of switching lamps and the capability of the circuit remote control.

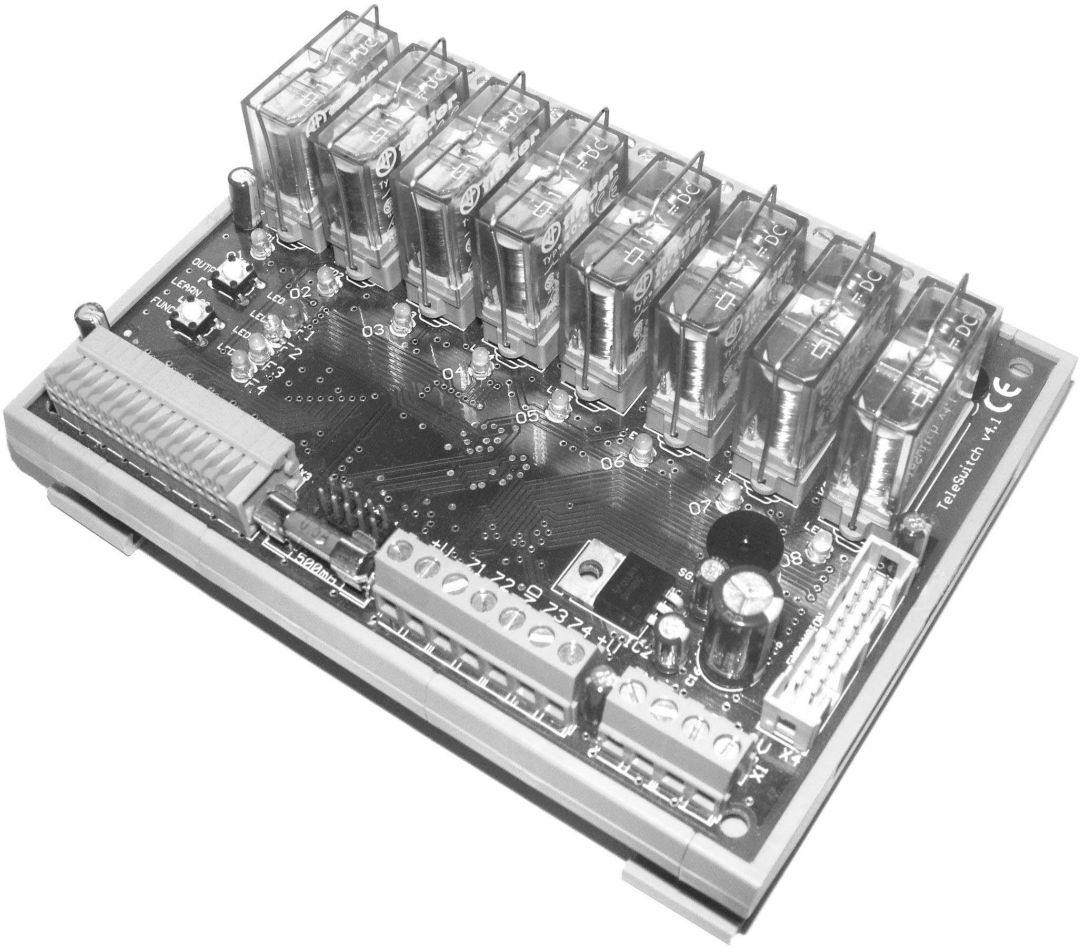


Figure 9 – Prototype device (Teleswitch)

The Teleswitch also needs to be very robust for the function it executes and to maintain the trust acquired by the user in operation of the traditional switch/lamp circuit. This also defines that the latency of the device should be as small as possible.

The cost of the device is also another requirement that limits the type and resources to be used in its implementation. In order to reduce the cost of the Teleswitch device implemented it can control up to 8 lamp circuits using power relays as it can be seen in Figure 9.

Among other tasks, the Teleswitch device is responsible for the binding between inputs and outputs as shown in Figure 7, reporting to the communication medium the corresponding events. The bindings in the device are initiated, by default, to direct control from inputs to outputs, but they can be changed manually (using two buttons on the device) or remotely. The Teleswitch allows the implementation of configurable reflexive actions between inputs and outputs in itself or in other networked modules.

The characteristic of reporting its actions through a network (input events, actuations, internal states, etc.), as well as the ability of receiving commands by the network interface, enable the possibility to be included in a much larger and higher level actions of an ubiquitous system.

This Teleswitch device can be used on home automation, doing stand-alone action like local switch lights on/off or controlling window shutters. But it can be a part of a system, capable of more complex actions that use context, e.g., the management of a light in a room in order to try to reduce power consumption or automatic light activation.

5.4.1 Teleswitch Hardware

The Teleswitch device hardware can be divided into 7 parts: input circuits, output circuits, processing unit, network interface; power supply circuit, local user feedback/control and expansion/programming circuit (appendix A1 - Teleswitch device schematic).

The input circuit consists in 8 digital inputs, plus 4 analog or digital inputs. The input circuit has electrical protection from short-circuits (using limiting resistors), voltage limiters (by using zenner-diodes) and high-voltage strikes (using varistors). The power that supplies the inputs is

protected with an independent fuse. The fuse status is reported to the microcontroller, allowing the analysis of the possible fuse burn-up.

The output consists in 8 relays (230v ac, 16A, 12V) that isolate the AC power from the controller circuit. The 8 relays are controlled using 2 integrated circuits ULN2803A (Eight Darlington Arrays). These integrated circuits were chosen because the price of the IC ULN2803 has almost no impact on the final cost of the device. Besides, the reliability increases when all the relays are in use.

The relays used have two contacts (normally opened / closed) that help to improve the power consumption for devices that are always on.

As a processing unit it was used an Atmel AVR microcontroller atmega329[1] that has a 32kbyte self-programming Flash Program Memory, a 2kbyte SRAM, a 1kByte EEPROM, a 8 Channel 10-bit A/D-converter, a programmable serial USART and up to 16 MIPS throughput at 16 MHz (appendix A3 - Atmel329 Datasheet (Microcontroller Features)). It has used a 32,768 kHz crystal to implement a real time clock (RTC) and to calibrate the microcontroller RC oscillator. The calibration of the RC oscillator is essential when using communication between devices. This microcontroller was selected because it is low cost, has firmware upgrade capability and adaptable features with several interrupt sources.

The network circuit uses a RS-485 differential line transceiver driver SN75LBC176DR[2]. This driver allows half-duplex communication in a bus between several devices. The receiving and transmission lines are connected to the TX and RX pins of the microcontroller serial USART. An extra IO pin from the microcontroller enables the data transmission in the driver. The receiving input is always active which helps a later detection of error during data transmission.

The network circuit, similar to the input circuit, also has protection against a data bus short-circuit, over-voltages and voltages strikes.

---

[1] Atmel - ATmega 329 - Product Full Description, AVR 8-Bit RISC Microcontroller with In-System Programmable Flash- http://www.atmel.com/dyn/resources/prod_documents/doc2552.pdf

[2] Texas Instruments - SN75LBC176 - Differential Bus Transceivers - http://www.ti.com/lit/gpn/sn75lbc176

The power supply circuit is very simple, as it consists only in a voltage regulator to supply the microcontroller with +5V from the external +12V dc. The noticeable feature is the capability of the microcontroller to monitor the voltage level before the regulator stops working. This monitoring helps the device to determine an imminent power supply shutdown or as a measure of the health of the external power supply. This information is used to save the device state before the power goes down.

The Teleswitch device also gives feedback to users by showing the activation of inputs or outputs with 8 leds positioned near to the respective relays. Other 4 leds are used to show the correct function of the devices and with combination with 2 buttons the users can alter the configuration by changing the default binding between local inputs and outputs. The feedback circuit also has a buzzer which indicates the user, with a soft beep, that the fuse was broken. This feedback helps the user to understand the state of the device and it can be used for debugging purposes.

Finally the Teleswitch device hardware is composed by an expansion connector that allows the connection to other sensors. This expansion is also used as an In Circuit Programming (ISP) for the microcontroller. For development, there is also available a JTAG connector.

5.4.2 Teleswitch Firmware

The Teleswitch prototype device software was programmed in C language, using a module structure to implement the different parts of the device firmware. No operating system was used because that would increase the overhead in the firmware using extra memory and processing power to execute the specific program. The memory usage in the firmware implemented was adjusted during the program development, and then defined as a fixed value.

The software processing was divided into 3 priority classes, which allow the distribution of processing. The processing classes consist in high-priority and timed processing, low-priority but timed processing and background processing.

The high-priority and timed processing class take advantage of several interrupt sources supplied by the atmega329 microcontroller. The processing executed in this class was reduced and optimized as much as possible, because the microcontroller can not do anything else while executing this class software segments.

The low-priority and timed processing class is based on interrupt sources of the microcontroller and its ability to have nested interrupt. This class software segments do not need to have very fast processing but they must not exceed the interrupt timer cycle to guarantee the necessary timing. The re-entrance of the interrupt and the possible stack overflow is not possible because the hosting interrupt is disabled during its execution.

The background processing is executed in the application main cycle. The segments that execute in background should avoid waiting loops to a particular interrupt processing. The processing of this segment should always start by checking if all the conditions for the processing are available. If not, they should wait for the next background loop cycle.

The firmware software of the Teleswitch prototype device is divided into the 10 modules enumerated below:

- Hardware Configuration and Setup;
- Inputs Processing;
- Outputs Processing;
- Binding Processing;
- Binding Configuration Management;
- Hardware Communication Management;
- Network Implementation;
- Real Time Clock (RTC);
- Manual Interface and User Feedback;
- Bootloader or Firmware Upgrade.

Hardware Configuration and Setup

The Hardware configuration and Setup module is responsible for defining the symbols to specific IO hardware pins, allowing the compatibility among different versions of the Teleswitch hardware. The configurations of the IO direction are setup by the module as well as the definitions of the CPU clock and timer cycle.

Input Processing

The Input Processing module is responsible for debouncing the switch data. This simple processing requires a high-priority and timed processing. This module also counts the transition of each input and generates the respective sensor events.

Output Processing

The Output Processing module interprets command events and executes them. The processing is buffered and the output is done in parallel and similarly to the output execution in industrial Programmable Logic Controller (PLC) devices.

Binding Processing

The Teleswitch device has the ability to bind the sensor events into output command events. That binding processing is implemented in the Binding Processing Module. This module processes the events that are available in a circular buffer, generating the eventual commands to the outputs. The processing is based on a table, called binding table. If the events exist in the binding, it executes the corresponding action.

Binding Configuration

The Binding Configuration management module is a group of functions that helps to manage the binding table. It is responsible for saving the table in a non-volatile memory and prevents the duplication of binding entries. It implements functions, such as: InsertEntry, GetEntry, RemoveEntry, LoadTable, SaveTable, ResetTable, etc.

Hardware Communication Management

The Hardware communication management module is responsible for converting the hardware RS-232 USART into the RS-485 communication. This module can detect data when it is being transmitted through the communication bus and only sends it when the bus is idle. When the transmission of data is initiated, it only stops when the transmission buffer is empty. This ensures that all the bytes are sent without gaps between them. All the outgoing bytes are also received and placed in the receiving buffer. The received bytes can be analysed later for possible collision detection. The state of the communication bus can be used by other modules.

Network Implementation

The Network Implementation module defines how the Teleswitch network operates. It implements the network architecture capabilities described earlier, i.e., the broadcast without collisions and the auto-address in a distributed way. A detailed description is presented in the next section.

Real Time Clock

The RTC module is responsible for generating a precise timer to be used in the automatic function of the outputs. This module also verifies, every second, the RC CPU clock speed and adjusts it, if necessary. This constant adjustment guarantees a better communication performance.

Manual Interface and User Feedback

The Manual Interface and User Feedback implements the routines that manage the local configuration of the device through the physical 2 buttons and displays several states using the LEDs in the Teleswitch device.

Bootloader or Firmware Upgrade

The Bootloader or Firmware Upgrade module is an independent program that is placed on the top of the microcontroller program memory and it is responsible for loading into the main programme memory a new version of firmware. It uses a very simple communication peer-to-peer protocol over RS-485. To be executed, it needs all the devices on the Teleswitch network to be blocked in order to communicate without errors with the server host.

5.4.3 Teleswitch Network

The Teleswitch prototype network implementation has the objective of showing that it is possible to implement a network of devices with the requisites proposed previously, with very little resources and at a much reduced cost.

The Teleswitch network implementation is divided into 3 layers:

- Physical Layer;
- Data Layer;
- Network Layer.

Physical Layer

The network hardware of the Teleswitch consists in a signal RS485 driver SN75LBC176DR connected to the USART microcontroller hardware. An extra microcontroller IO is used to activate the transition of the RS-485 driver. The physical signal is very robust to electromagnetically noisy environments, because it uses a differential balanced line in combination with a twisted pair wire.

The reception circuit of the RS-485 driver is always active, allowing the microcontroller to send and receive at the same time and this way it enables a later data collision detection. The bus speed was set to 115.2kbps which is fast enough for sending home events (light switching or movement detection, etc) and slow enough for allowing the execution of nearly 700 instructions between receiving bytes interrupts on the microcontroller that runs at a 8 MHz clock. A faster CPU speed was not used because it requires more expensive clock circuit hardware and increasing device final cost.

Data Layer

The data layer is responsible for receiving and sending the bytes from and to the RS-485 bus. It may detect some physical errors like: data overrun and framing errors. These errors are directly obtained by the USART controller of the atmega329. The data integrity is checked in the network layer.

This layer is capable of sending a full packet of bytes without any pause among them. In order to do this, it uses a send buffer, capable of containing a full network packet and it also uses a double bytes transmission buffer, supported by the USART.

The activation of the RS-485 driver is done immediately after the writing of the first byte in the USART transmission buffer. The deactivation of the transmission needs to wait for the output of the last bit to the bus. This information can be acquired by looking at the USART transmission complete flag. The Atmel microcontroller can mask this flag, generating the respective interrupt, which is ideal to this application because one can use the CPU to do other tasks during the transmission of data.

The data layer also detects the state of the bus which is used for the implementation of Carrier Sense Multiple Access (CSMA) by the network layer. The bus can have the following modes: Idle Mode, i.e., no signals on the bus; Receive mode, when it is receiving data from a remote device; and Transmit Mode, when sending data to remote devices.

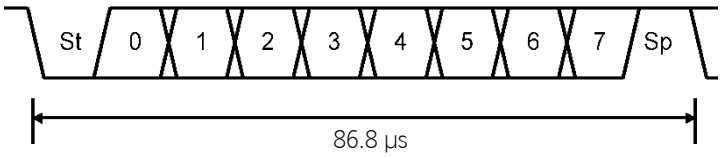


Figure 10 – USART transmitted byte at 115.2kbps (Start Bit, 8 bit data and Stop bit)

The bus starts on idle mode and goes to receive mode when the bus has activity. The USART from the ATMEGA329 only has a complete byte received flag, which only appears 10 bits after the first activity on the bus, as shown on Figure 10, resulting in a delay of 86.8µs in the detection of activity on the bus. The delay increases with slower transmission speeds. This delay has a real implication on the CSMA algorithm because it is not possible to detect the carrier sense of the bus effectively.

To minimize the lack of a receiving in progress flag from the USART, it has been used an external interrupt - Pin Change Interrupt (PCINT). The PCINT interrupt allows an interrupt to be generated when the signal changes the state on the USART RX pin. This interrupt is immediately deactivated when it detects any activity and is enabled when a complete byte is received. This procedure is done to prevent an interrupt on every change in the receiving bus bits and this way it saves some CPU clock cycles.

All the bytes received from the bus are placed on the receiving buffer for the network layer for later processing. The transmission bytes are also received and used for collision detection that is implemented in the network layer.

The receiving mode ends when bus activity is not detected during a time of one and a half bit (13μs at 115,2kbps), counting from the last complete byte received.

The transmission mode starts when no activity on the bus is detected and ends when the transmission buffer is empty. When a collision happens, the transmission does not stop, as opposed to the CSMA/CD algorithm, because of the extra CPU usage needed to check all the output data during the transmission process.

Network Layer

The network layer is responsible for preparing the packages of data, verifying the integrity errors of the received package. This layer establishes the link between the data layer and the application layer.

The medium access implemented in this layer can be done using three different methods, depending on the state of the network or on the requirements of the application layer. The three methods are: carrier sense multiple access (CSMA), temporary medium reservation and network bus lock.

In the CSMA, a Teleswitch device that wants to send data must first verify the absence of traffic before it starts transmitting. This method guarantees that an outgoing transition is not corrupted by another device packet, but it does not prevents data collision if two or more devices want to send data at the same time. This limits the usage of this method only when using in protocols that require acknowledged packets, i.e., applications that ask information and wait for a reply.

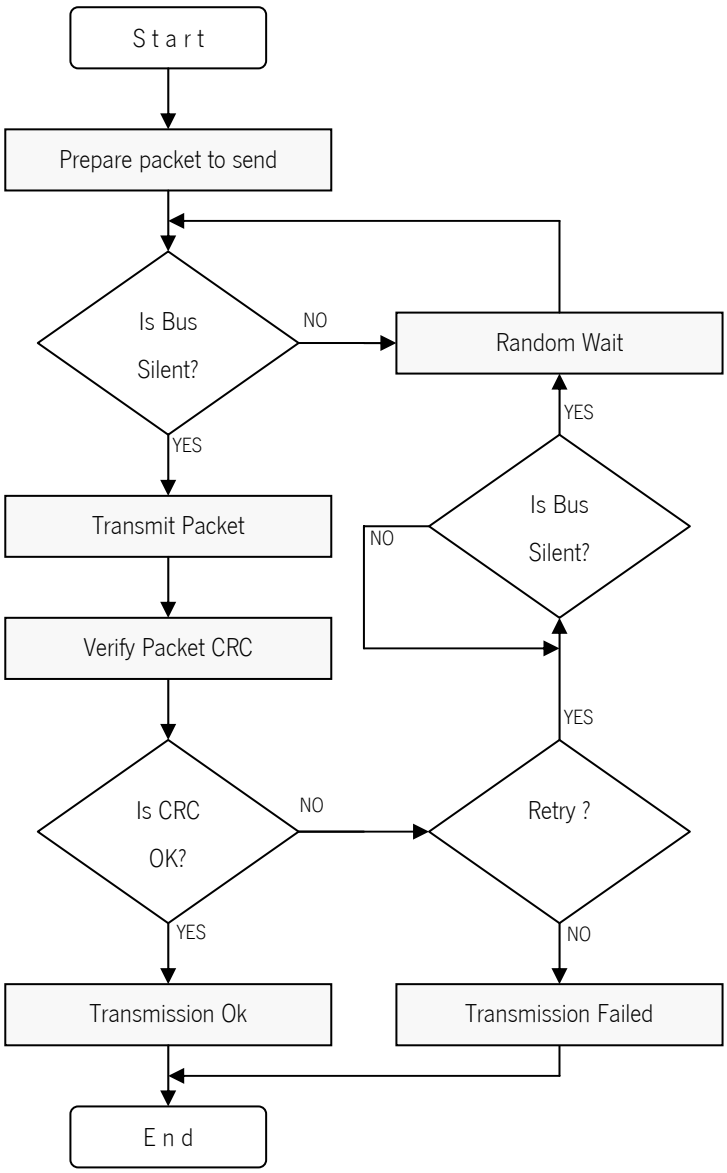


Figure 11 – Teleswitch Network CSMA algorithm

This method is ideal for centralized applications or master/slave architecture. It is used when a management device like e.g. a communicator device (where the data and hardware management function are implemented) does the monitoring of the active devices and whenever a change of configuration is needed.

The collision detection is also implemented in this layer. The data collision is verified after each complete packet is sent, as shown in Figure 11, using the loopback capability of the data layer. Only the transmit devices know if a collision happens, the other devices only receive a packet with an invalid CRC[1]. Retransmission after a collision is executed after a pseudo-random wait which helps preventing consecutive collisions.

In the temporary medium reservation, a device can transmit to the data bus without sensing the medium during a slice of time. During this time, the medium is reserved and only one device has permission to transmit. Therefore, no collision happens. This method is used when a device needs to broadcast information to several devices without requiring the acknowledge reply. It is used to broadcast events (sensor events and feedback events) to all devices on the Teleswitch network. The receiving devices can use those events to generate other local actions or they can simply register that information on a database for later context-aware building.

The temporary medium reservation has two stages: reservation request and medium reserved. The reservation request is done by broadcasting a request packet with a pseudo-random ID using the CSMA algorithm and waiting for a reply from a Medium Access Control (MAC) coordinator device with an authorization correct ID. When the correct reply is received, the device starts sending packets to the medium during a reserved time slice. If no coordinate reply packet appears, after a timeout, the device broadcasts a self-reservation packet and become the MAC coordinator. Any device on the net can turn into a MAC coordinator. To prevent the appearance of more than one MAC coordinator on the network, any packet received from a coordinator with a different address from its own, instantly causes the demise of the coordinator function.

[1] The cyclic redundancy check (CRC) is used as a checksum to detect accidental alteration of data during transmission. More information available on-line in: http://en.wikipedia.org/wiki/Cyclic_redundancy_check

Any device that receives a MAC reservation request packet automatically stays silent for a period time, defined as coordinator timeout reply plus the reservation slice time. When an authorization packet from a MAC coordinator is received, this back-off period is adjusted to the time slice interval. In this method, a packet collision can happen during the medium reservation request stage, but it is greatly minimized because of the two back-off conditions.

This method permits the implementation of a distributed network and is ideal on a network in which the CRC errors are mainly provoked by packet collisions.

Finally, in this network layer, it is implemented the medium lock. It consists on blocking the communications, i.e., the devices do not transmit and ignore any data until they receive the unlock packet (an escape sequence of bytes).

This method is used to make the upgrade of the firmware in devices and was implemented since the size of the boot space in the AVR microcontroller was not enough for the network stack.

In order to lock the medium, the first step to be made is to broadcast a lock packet; optionally an echo request can be broadcast to verify if all the devices are locked. Next, an unlock packet is sent to the particular device that access is wanted. After that, it is possible to request any specific application with different network protocols and even different transmission speeds. To unlock the bus first, it is important to put the unlock devices in the correct state and then broadcast the unlock packet to the net to unlock all the remaining devices.

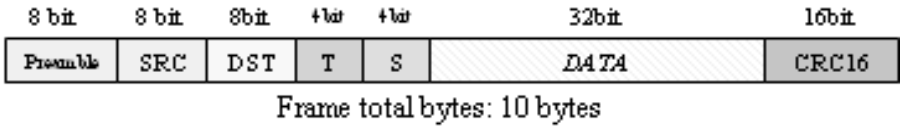


Figure 12 – Teleswitch packet frame format

The frame structure used on the Teleswitch net has a fixed length (10 bytes), as shown in Figure 12. It consists in: Preamble, Src Address, Dst Address, Type of the frame (T), Sub-Type (S),

Data and ending with a standard CRC16 (appendix A4 - Teleswitch Network (common definition file)).

The preamble is used to synchronize the start of the receiving frame and is 8bits because the Atmel USART used is byte-oriented. The value of the *preamble* is 0xA5 which creates a square signal, shifting the phase in the middle of the receiving byte and therefore, improving the detection of data collisions.

The *src* (source address) field has a length of 8 bits and appears before the destination address field, as apposed to other network protocols like e.g. Ethernet frame. It aims to improve microcontroller memory and code efficiency. Next is the *dst* (destination address) field. The value 0xff is reserved as broadcast address and all address ending with a 0xf is used to send packets within a group. Because of these addresses assignments, this network protocol can address a maximum of 240 devices. The address of 0xfe is reserved to devices which are not yet configured and the range of [0xf0..0xfd] should be used to devices that make bridges between protocols or devices that manage or monitor the device network.

The frame type is defined in the fields: Type (T) and Sub-Type(S). The frame type or packet type has four main values, which are: MAC, control, events and binding. The frames with MAC type are used to control the access medium in the temporary medium accesses method described earlier. The control frames enable the remote management of the device by transmitting information related with the hardware, network statistics and special functions like: reboot, firmware boot-loader, etc. The event frame is sent when a device needs to broadcast actions and events. The binding frame is used to manage binding table entries (configuration of the reflexive actions).

The *data* field has a length of 4 bytes and it is used to send generic data related to the type of the frame. Finally, a standard *CRC16* field was included to guarantee the data integrity and to allow the collision detection.

The address assignment can be done automatically using a similar strategy as used on the AUTOIP in the TCP/IP networks. This strategy starts by selecting a random address, then, it broadcasts a packet that queries the use of that particular address. If no device responds, it

assumes that the address is free. If the address is already in use, the device receives a reply and starts the process over again. These messages are always sent in broadcast mode and with a medium temporary reservation access method.

5.4.4 Sensor Events and Actions

The capability of sending events in broadcast to other devices is essential to make possible the creation of a system with functions and reactions that have not been planned in the initial design, i.e., a future device can grab the information that travels on the bus and do other functions, creating more sophisticated applications.

In the Teleswitch prototype, the events are the input from the user, e.g., when a user switches a control in the hall, this generates events informing: the input that was activated (the state of the switch) and how many transitions the user has done. That information is expelled from the device and one or many Teleswitch can grab that information and do a specific action.

Table 7 – Teleswitch data bus activity after light switch action from a user

Teleswitch Data Bus Activity				
#	SRC	DST	TYPE	OBS
0000	0x10	0xFF	MAC	REQUEST ID:0x1452
0001	0x11	0xFF	MAC	AUTHORIZED ADDR:0x10 ID:0x1452
0002	0x10	0xFF	Event	LOCAL IOId:5 IN:5 Type:KEY Bind:PRESSED Data:0
0003	0x10	0xFF	Event	LOCAL IOId:133 OUT:5 Type:RELAY Bind:ON Data:0
0004	0x10	0xFF	MAC	REQUEST ID:0x14DB
0005	0x11	0xFF	MAC	AUTHORIZED ADDR:0x10 ID:0x14DB
0006	0x10	0xFF	Event	LOCAL IOId:5 IN:5 Type:KEY Bind:PRESSED TRANS:1 Data:0

The Table 7 shows the packets decoded that result after switch action from a user. The device with the address 0x10 starts by requesting a medium temporary reservation, with ID 0x1452. The device 0x11 (the MAC coordinator at that moment) gives the authorization for medium access to the device 0x10, by replying with a broadcast packet that identifies the device address and the

correspondent request. The device 0x10 starts by broadcasting the user event (pressed key on the input 5 (IOId 5)) and immediately transmits the local reflexive action which, in this case, is the action of turning ON the relay 5 (IOid 133) on the device. Because the input sequence calculation on Teleswitch ended after the MAC time slice, the device has to request the medium access again. After it receives the authorization, the device 0x10 broadcasts that the user remains with the input pressed and it only generated one transition in the switch.

Sensor Event

In the Teleswitch device, the sensor events are the occurrences captured by the digital and analogue inputs. Each event is identified with a value of 8 bits that is directly associated with a physical input or output. The most significant bit defines if the event is a sensor event or a feedback event.

For each event there is also the binding data that holds the information related to what happens to the specific IO, e.g., to a digital input the binding data is the state of the input (high/low or Pressed/Released) and the number of times the input changed.

The event identification and the event binding data token describe the events in the Teleswitch network.

Feedback Event

In the Teleswitch device, the Feedback Event is generated by changes in the relay outputs. In the case of the relay output, the binding data of the event can only be 0 or 1 (relay off or relay on). Although the action of executing in a relay can not only be turn ON or turn OFF, it is also possible to request a relay state toggle or pulse activation.

The 8 bit from the event identification in combination with the extra 4 bits of the event binding data result in a total of 2^{12} (4096) different events that can be generated by a single device.

5.4.5 Event Binding

The binding table consists in token event/action. Each device has a table that maps a specific event to an action in local outputs. The limitation to execute action only in the local device outputs is related with the low memory resources available, but this does not mean that the execution can not produce actions in other devices. These can bind the feedback action to change an output.

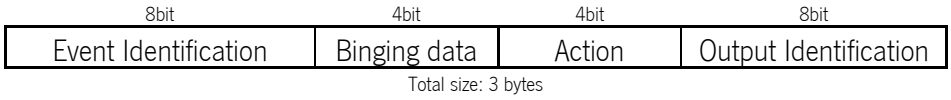


Figure 13 – Teleswitch binding table entry format

An entry on the binding table is defined as shown in the above figure. The Event Identification defines the IO address of an event. The Binding Data is a fixed data that is related with the type of IO, e.g., if the IO is an input, the Binding data contains the transitions counter and the final inputs state. The Action field defines the function that is executed in the output. That action depends on the type of the output. The Output Identification defines the output where the action is executed.

The Teleswitch device has a total of 256 cells in the binding table which uses 768 Bytes of RAM (around 38% of total RAM available). In order to bind with events from other devices in the network, a table that contains the remote addresses points to the binding table entries, as represented in Figure 14. The first entries are for the local device, which improves the local processing.

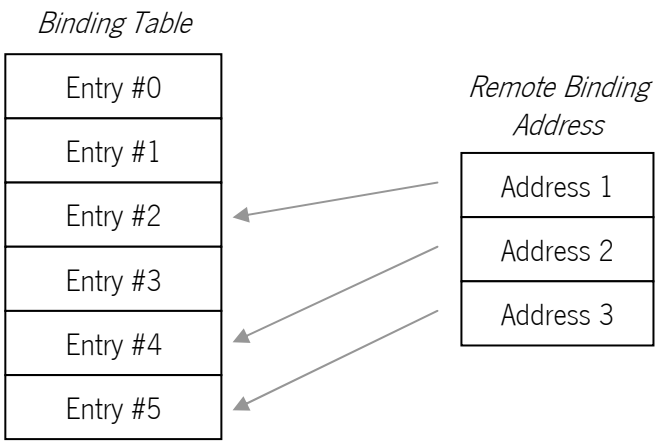


Figure 14 – Teleswitch binding table entry format

The group address can also be used in the remote binging address table which reduces the usage of entries in the binging table.

The binding table defines the basic behaviour of the Teleswitch device. The combination of all the tables of the devices in the network defines the overall basic system behaviour.

Table 8 – Example entries in the Teleswitch binding table

Events	Actions
Simple pulse switch control	
Button pressed (Input 0, Transition to up)	Light state toggle (Output 0)
Windows Shutter control	
Up Button Pressed (Input 1, Transition to up)	Up control line, toggle state (Output 1)
Up Button Pressed (Input 1, Transition to up)	Down control line, off-state (Output 2)
Down Button Pressed (Input 2, Transition to up)	Down control Line, toggle state (Output 2)
Down Button Pressed (Input 2, Transition to up)	Up control line, off-state (Output 1)

5.4.6 Monitoring and Status

The Teleswitch device allows access to some internal variables related with: inputs/outputs (actual state of each IO); network metrics (processed packet, total CRC errors, buffer overflows occurred); hardware (power supply voltage and hardware fuses status) and software parameters (firmware version, stack low water marker and number of reboots). The availability of this status allows a monitoring station to preview and determine any possible system failure.

5.5 Summary and Conclusions

In this chapter, an approach to create an environment that is configurable and able to react automatically was presented.

The biological ANS reflex arc process was used as an inspiration, because it has analogous characteristics needed in the autonomous ubiquitous system. The events created by the sensors and events from the actions in combination with the ability to bind them with actions create a structure that allows the implementation of basic behaviour.

Sending data using events and broadcasting it to the communication medium permits the interoperability with new devices, because new devices can catch information in communication media to implement new functionalities. The monitoring of the system is also simplified since all info is available in the bus and there is no need to request data on regular bases.

The approach adapted has implications on some network features and architecture. The Network implementation needs to be able to send data in broadcast mode, avoiding data collisions and work as distributed as possible without the usage of centralized services. The protocols implemented must also be very simple as they need to be embedded in devices with very low-resources.

The Teleswitch prototype device was created to validate the concept presented earlier in this thesis. This prototype device shows that even with very low-resources it is possible to implement the basic and essential functionality and at the same time, it allows the information capture to be reported and used in more sophisticated applications.

In the next chapter, a case study of a real installation is presented and in the following chapter the information to be emitted by the environment devices is processed, using from simple statistics to more complex artificial intelligence techniques, in order to create context-awareness applications.

6 Case Study: Home Environment Application

In this chapter it is presented a case study using the system architecture adopted in this thesis. The case study consists of the instrumentation of a real house where its inhabitants were not asked to change their daily routine, i.e., the study of a real family in a real life home scenario.

6.1 Overview

The home environment is an ideal place to test the ubiquitous system appropriateness because in this atmosphere the routines of the inhabitants of everyday life inevitably happen. Therefore, the capability of such system making a positive intervention and impact can be tested.

In order not to create any interference in the normal behaviour of the inhabitants of the flat, no extra tasks were imposed to them, i.e., the users were not required to report, using a log-like register procedure, what they were thinking, doing or intending to do. This aspect was very important because the experiment is supposed to be extended for as much time as possible (several years) in order to be able to detect changes in behaviour over long time intervals. Any requests for inhabitants' extensive reports inevitably limit the motivation of the users to be a part of the study and may have impact in their normal daily life behaviour.

In a home environment, a system that is constantly used is the illumination system. This is used by the inhabitants all the time and this usage reflects the users' habits and behaviour, making the house illumination system a good case study.

The type of housing used in this case study is an apartment with a total area of 225 m^2, having a total of 12 divisions consisting in: a suite bedroom, a bedroom, a study room, a living-room, two WC, a kitchen, a corridor, a hall, an entrance and two balconies. The plan of the flat is

presented in Figure 15. This flat is the home of 3 people: a man, a woman and a little child and it is used every day.

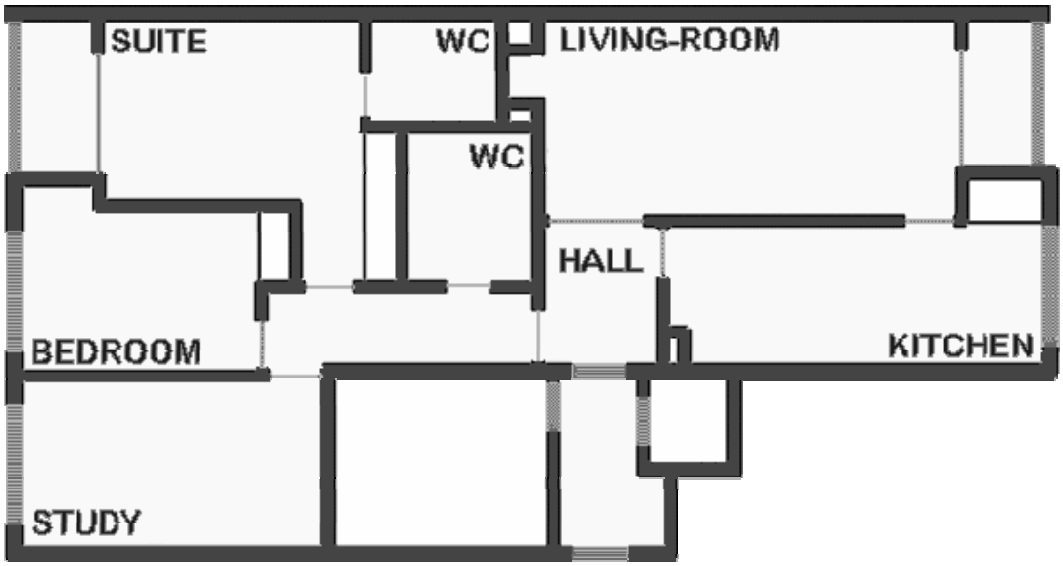


Figure 15 – Case Study, Flat plan

6.2 Objectives

The case study implemented was created to test the philosophy previously presented in this thesis, putting into practise the requirements, characteristics and orientations presented in the development of a ubiquitous system.

The main objectives when implementing the case study are:

- To test the robustness – the robustness and reliability of the system designed is examined, i.e., the distributed network of devices that control the lighting system has to guarantee that the basic functionality is always available. The system is supposed to be placed in between the traditional light electrical circuit and therefore it must be as robust and reliable as a standard circuit. The basic functionality can not be

affected by failures of the higher level system, that monitors and provides the feedback to the users;

- Testing of the non intrusive design – the case study is focused on an installation that is as unobtrusive as possible, from the user point of view. These systems are supposed to cause minimal impact in the environment, in particular in terms of visual implications as well as functionality. Despite minimal impact being an objective, users may take advantage of some extra capabilities that the basic system can provide, such as, switching all lights on\off when leaving the house, controlling of lights from different switches.

- data capture for analyses – the installation create an automatic data capture system that gathers information which can be used for later analysis and implementation of new awareness applications and functionalities in the home environment;

- adaptability to context – the usage of a simple light system is the basis to explore the capability to detect patterns from users and to execute useful actions in the home according to the context;

- suitability to home environment – some basic useful, but not conventional functions of the lighting system are setup and the adaptability to those small changes by the users are investigated;

6.3 Installation

In the installation of the system, 30% of the apartment's electrical standard electrical circuitry was changed. The basic changes made in the circuit go from the light switches to the light bulbs derivation electrical boxes. The Teleswitch device is used as controller unit of the lights. The data that is captured, broadcast to the data bus and monitored by an IP communicator which is

responsible for establishing a bridge between the Teleswitch data bus and the TCP/IP network. The communicator device is presented in the next sections.

The control and feedback of the system is available in a PC with a tactile monitor and a PDA with wi-fi connection.

6.3.1 Characteristics (Inputs / Outputs)

The installation of the system in the light system uses the light bulbs as the outputs in the environment; and light switches as well as Passive Infrared (PIR) motion detectors as inputs, distributed all over the flat as presented in Figure 16. The apartment has a total of 20 light points (actuators), as indicated in the Table 9 and a total of 27 light switches (sensors) in 18 different locations. The flat light switches have the toggle design and the connection appears in two types: SPST and SPDT[1]. In the case study installation, the switches needed to be of type SPST and therefore the SPDT was easily converted to SPST type by connecting only one way of the switch.

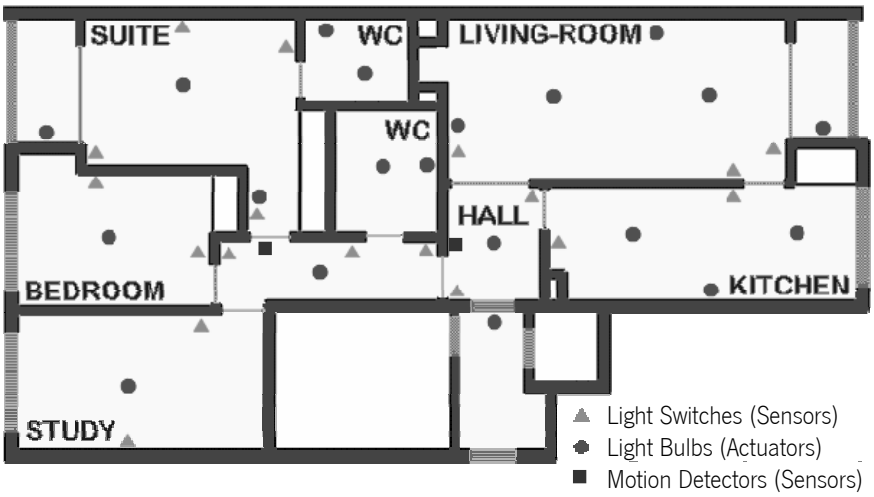


Figure 16 – Case Study, Flat apartment plan with sensor and actuators distribution

[1] SPST (Single pole, single throw); SPDT (Single pole, double throw). More information available on-line at: http://en.wikipedia.org/wiki/Switch

Table 9 – List of lamps, with output power, in the case study flat apartment

Flat Division	Light Point	Power (W)	Description / Obs.
SUITE	Main	35W	Main light bulb
	Entrance	20W	Entrance & Ambient light
SUITE WC	Main	10W	Main light bulb
	Mirror	50W	Mirror Light
NORTH BALCONY	Main	40W	Exterior light
BEDROOM	Main	50W	Main light bulb
STUDY	Main	40W	Two fluorescent lights
CORRIDOR	Main	100W	Two light bulbs
MAIN WC	Main	10W	Main light bulb
	Mirror	100W	Two Lights
HALL	Main	50W	Light Bulb
ENTRANCE	Main	40W	Light Bulb
KITCHEN	Main	15W	Main light bulb
	Sink	40W	Two fluorescent lights
	Laundry	15W	Light bulb
LIVING-ROOM	Main	50W	Light bulb
	Dining Table	140W	Four light bulbs
	Environment	35W	Environment Light
	Indirect	100W	Indirect light
SOUTH BALCONY	Main	40W	Exterior light

6.3.2 System Architecture

The system architecture can be divided into three main components: the basic functionality component; the system gateway and enhanced functionalities component; and the user control and feedback component. The structure of the installation is presented in Figure 17.

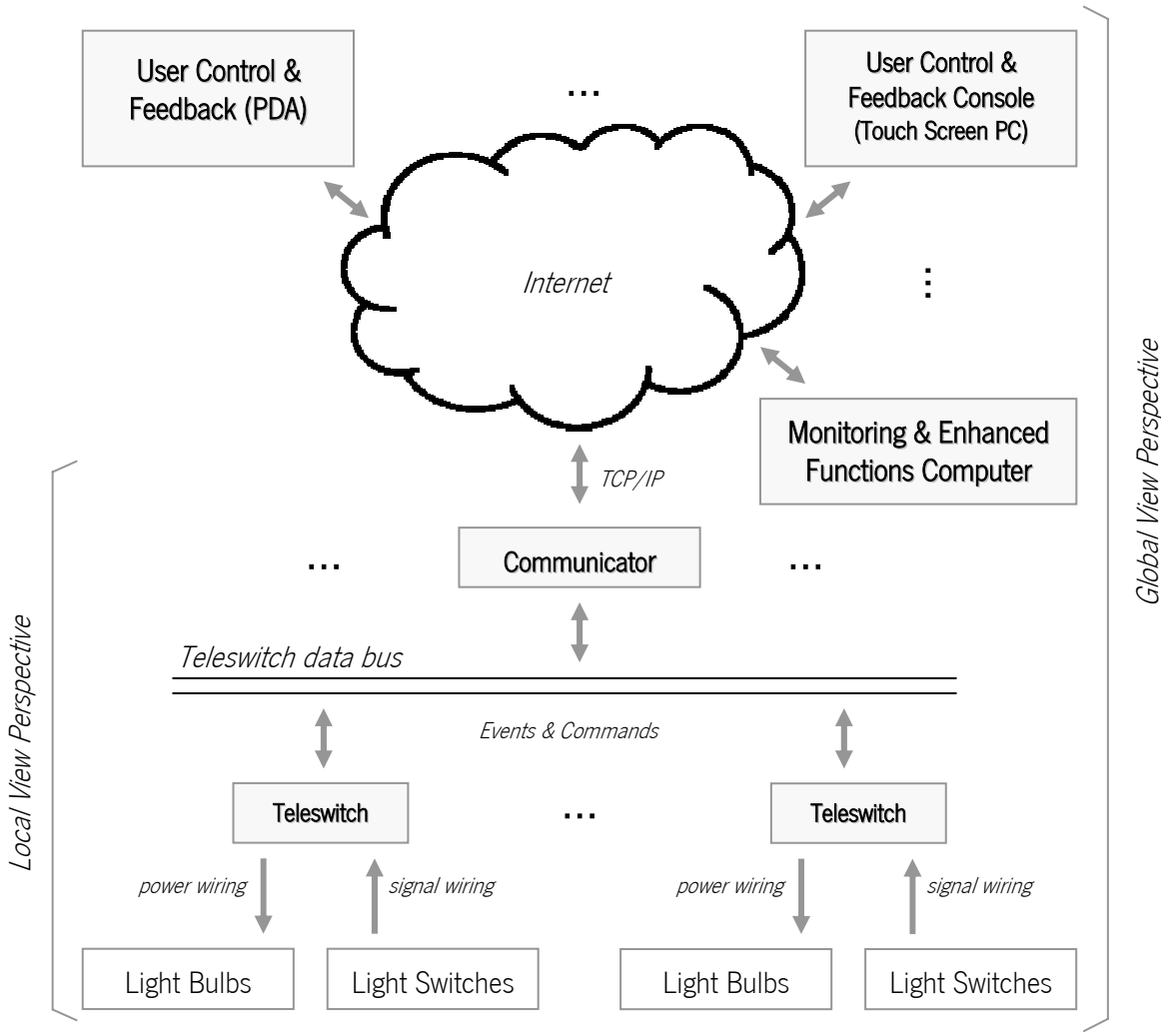


Figure 17 – Case Study installation component structure

The basic functionality of the components (Teleswitch devices) are responsible for reproducing the standard operation of the electrical illumination system, i.e., the basic light switching and the ladder switching type (turning on the light in one switch and turning it off in

another). The basic functionality component also implements some non-standard basic functions, like switching all the lights off in a single switch or controlling the light intensity. This component implementation is based on the Teleswitch which is also capable of being reconfigured by other system components.

The system gateway and enhanced functionalities component implement the data bridge between the basic component user control and feedback component. This component is also responsible for monitoring and registering, in a DB, the events that travel in the Teleswitch data bus. In this component there are also implemented some functions that demand more resources and can not run in the basic functionality component, e.g., the statistical computation or AI learning algorithms. This component is implemented using a communicator device, that is presented later on, and a personal computer (PC), although the implementation can use more than a communicator and/or PC.

Finally, the user control and feedback component is responsible for user interface to remotely control and monitor the state of the system. This component uses a PC with a touch screen as the user interface or alternatively the user can run an application on a PDA with WI-FI connection. The system information displayed is supplied by a gateway component. The details of this component are explained in this chapter.

6.3.3 Basic Functionality Setup

As indicated earlier, the basic function is based on the Teleswitch device. In this installation, in order to control 20 light bulbs it was necessary to use 3 Teleswitch devices. These devices, together, provide a total number of 24 relay outputs, 48 digital inputs and 768 configurable binding entries.

In order to take advantage of the resources available in the devices, some light bulbs were connected using a simple half-wave rectifier which allowed the implementation of three light levels output (light off, half light, and light on).

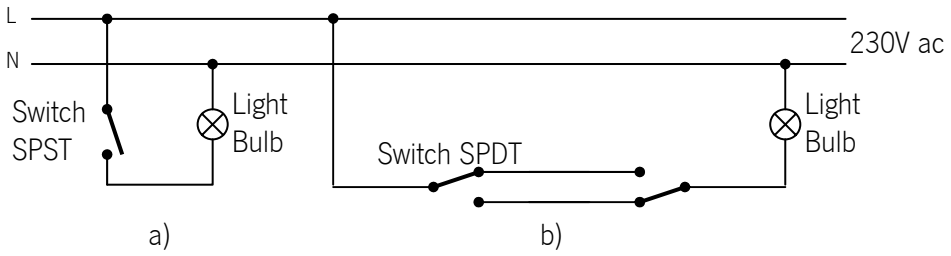


Figure 18 – Typical light bulb circuit connection: a) simple light switching; b) XOR light switching

In Figure 18 it is shown two common light bulb circuits included in a home. In the circuit a) the light bulb state is the same as the state of the SPST switch and, therefore, the light state can be determined by simply looking at the switch. In the circuit b) that determination can not be done looking only to the state of one switch because the light state depends on the combination of two switches. To simplify the connection of the circuit b), the XOR operation is performed by Teleswitch and the SPDT switch is used as a SPST switch. The resulting Teleswitch binding table is presented in the Table 10.

Table 10 – Teleswitch Binding table: simple light switching; XOR light switching

Events	Actions
Simple light switching using a SPST type switch	
Switch On (Input 0, Transition to up)	Light state On (Output 0)
Switch Off (Input 0, Transition to down)	Light state Off (Output 0)
XOR light switching using two SPST	
Switch 1 On (Input 1, Transition to up)	Light state toggle (Output 1)
Switch 1 Off (Input 1, Transition to down)	Light state toggle (Output 1)
Switch 2 On (Input 2, Transition to up)	Light state toggle (Output 1)
Switch 2 Off (Input 2, Transition to down)	Light state toggle (Output 1)

In the Teleswitch the configuration of the simple light switch has to take into account the switching from a remote place. The remote actions can change the light state and consequently this can be different according to the state of the switch. To solve this problem, the binding table was configured to toggle the light state on each switch change, as presented in the Table 11.

Table 11 – Teleswitch Binding table for simple light switching (toggle light state)

Events	Actions
Simple light switching using a SPST type switch	
Switch On (Input 0, Transition to up)	Light state toggle (Output 0)
Switch Off (Input 0, Transition to down)	Light state toggle (Output 0)

The Teleswitch was placed in the electrical derivation boxes that are present above the division doors all over the flat. The electrical derivation boxes are the ideal place to locate the Teleswitch devices, because it is the middle place in the electrical light circuit, i.e., the light bulb power cables and light switch cables are connected.

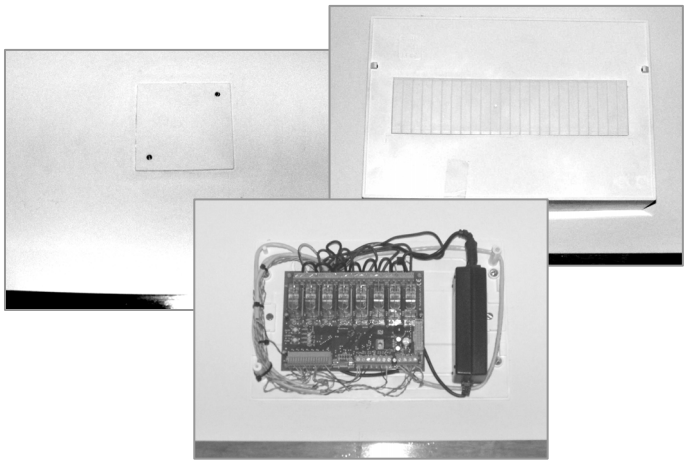


Figure 19 – Teleswitch installation over the electrical derivation box

The existing light switches in the apartment have not been changed, maintaining the switches and therefore making the installation more unobstructive to the normal habits of users. The only visible change in the flat was the presence of the Teleswitch installation boxes, as shown in Figure 19.

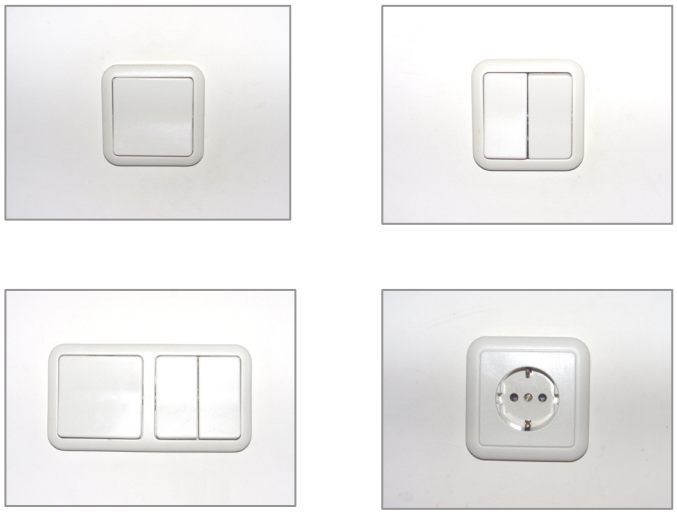


Figure 20 – Flat switches and plugs (before and after the system installation)

6.3.4 System gateway setup and enhanced functionalities

As a system gateway it was used the communicator device presented in Figure 21. This device is an embedded system, programmed in C language, with low power consumption, which main objective is the execution of network services.

The communicator device is based on the microcontroller IP2022 from Ubicom with internal 64kbytes of flash, 16kbytes of PRAM, 4kbyte of ram. The board also has an external 128kbytes of SRAM (on board bottom side) and an external Atmel flash of 512kbyte in size. This device, beside of the Ethernet connector, also has an RS485 interface, a RS232 connector, a X10 RJ11 connector, including 4 digital inputs and 4 isolated digital outputs (appendix A2 - IP Communicator device schematic).

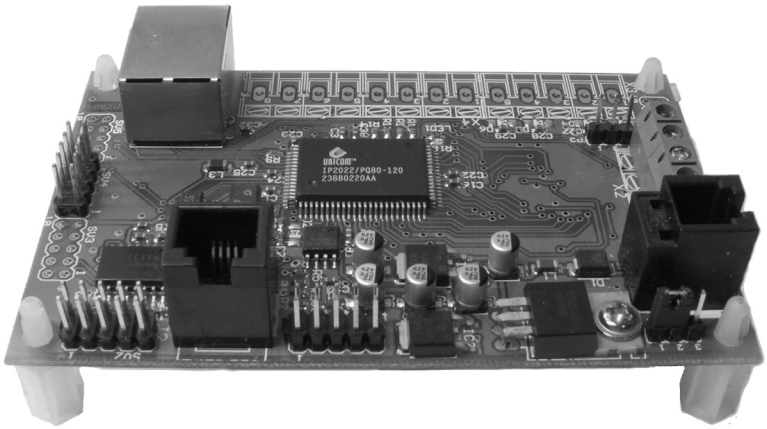


Figure 21 – Communicator device (based on Ubicom IP2022)

The software of the communicator is divided into five network main services, as presented in Figure 22, that allow the high level application to send and receive information to the Teleswitch network. The services implemented are:

- Event Broadcast (via UDP): this service broadcasts to all nodes in local network the events that are transmitted over the Teleswitch data bus. It allows any monitoring server, tactile console or device to receive the feedback information. These services reflect the system philosophy through which all devices emit the events to all the others that may later appear;

- Event Monitoring (HTTP client): this service monitors the Teleswitch devices, time-stamps them and sends them directly to a HTTP server. This service was used by a server that was programmed in PHP, which registers the events into a mySQL database. This notification service permits the usage of a server that is not connected to the local network;

- Teleswitch Bridge (telnet): the telnet bridge to the Teleswitch data bus allows the direct access to the devices RS485 data bus, allowing the execution of applications

that may use different communication protocols. The firmware upgrade of the Teleswitch device is an example of an application that needs that low-level data bus access;

- CGI[1] (HTTP Service): this service is the API that allows the configuration of the communication support CGI and the interaction with the Teleswitch devices CGI. The communication support CGI (/sup) permits the configuration of the communicator network parameters (ip addr, ip mask, dns, ntp server, etc). The Teleswitch CGI (/swt) has three main functions: send the actions to the devices; request input/output stats; and allow the sending of raw Teleswitch data frames. This service is used by the control and feedback tactile consoles and the PDA.

- Web HTTP service: the http service stores and serves the configuration interface of the communicator. The XML configuration of the system can also be served by the Web HTTP service provided by the communicator;

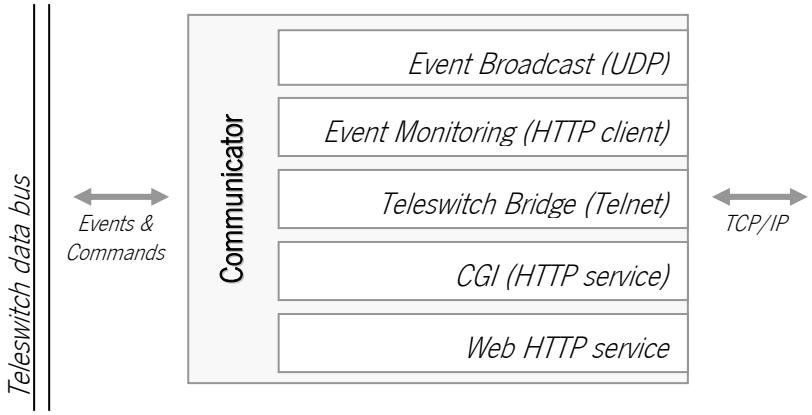


Figure 22 – Communicator device network services

[1] Common Gateway Interface (CGI)

Some other features implemented are also: ntp client (network time protocol) used to get the correct time for stamping the events; Wake-by-Lan service to request the power on of the feedback console device and finally the communicator firmware upgrade functionality.

6.4 User Interface (control and feedback)

The user interface is an installation component that allows users to control and visualize the light state of the case-study home. The component was created to be as adaptable and flexible as possible, by using web technology (html, java script, etc). As a description file, an Extensible Markup Language (XML) format has been selected which is a universal format because it allows the definition of the mark-up elements by the users. In Figure 23, it is possible to see the installation of the console placed in the hall of the flat.

Figure 23 – User Interface console installation (control and feedback through a touch screen)

6.4.1 Configuration file (XML file description)

The xml file is used, by the touch screen console and the PDA application, as the configuration file that holds the information about the actions to be executed in the flat, as well as the labels and graphical information to be used in the display (appendix A5 - UDOMUS XML Configuration File).

The xml description starts with the global tag <DOMUS> that holds the description of the home automation elements. The usage of the XML attributes was avoided because this feature reduces the ability to add new sub-tags, making the new xml description incompatible with older versions.

The <DOMUS> description is divided into three class elements: <SYSTEMS>, <COMMANDS> and <SPACE id=...>.More tags can be added later for more detailed specification or to manage some special requirement of a specific technology, e.g., heating systems, security, remote control access.

The <SYSTEMS> tag contains the global definition of the control system present in the installation. In this case study, the control system is the <Teleswitch> and the variable needed by the console devices is the URL of the gateway, as presented below:

```
...
<SYSTEMS>
        <TELESWITCH>
            <GATEWAY>
                <URL><![CDATA[http://192.168.0.99/]]></URL>
            </GATEWAY>
        </TELESWITCH>
    </SYSTEMS>
...
```

The <COMMANDS> xml tag holds the global definition of the commands that can be executed by the user. In the case study, this tag defines the images that represent the state of the command and the correspondent power percentage.

```
...
<COMMANDS>
        <TYPE>
            <ON>
                <IMAGE>img_sup/bt_on.gif</IMAGE>
                <AIMAGE>img_sup/bt_on_r.gif</AIMAGE>
                <OIMAGE>img_sup/bt_on_g.gif</OIMAGE>
                <POWER>100</POWER>
            </ON>
            <HALF>...</HALF>
            <OFF>...</OFF>
        </TYPE>
    </COMMANDS>
...
```

The <SPACE> tag defines the command per compartment of the flat. It can also be a group of rooms or a global command of the flat. This tag has an 'ID' attribute that helps to identify the space definition. The <SPACE> tag contains sub-tags that are related with the users interface graphical representation, such as: <TITLE>, <IMAGE>, <POSITION>, <COORDS>; and the <ACTION> tag that defines the action that can be executed in that space.

```
...
<SPACE ID="WC">
        <TITLE>WC</TITLE>
        <IMAGE>img_home/wc.JPG</IMAGE>
        <ACTION>...</ACTION>
        <ACTION>...</ACTION>
        <POSITION><![CDATA[left:170px; top:248px;]]></POSITION>
        <COORDS>173,248, 258,248, 258, ..., 173,248, 173,248</COORDS>
</SPACE>
...
```

Typically, the <ACTION> tag defines the action per light point in the flat. In this tag it is defined the action title (<TITLE> sub-tag), the system where the action is executed (<SYSTEM> sub-

tag), the feedback input/ouput address which is the combination of the Teleswitch address and IO address, and finally the execution command <COMMAND> which holds the CGI and parameters to execute in the gateway. An example of an action definition is presented below:

```
...
<SPACE ID="WC">
    <TITLE>WC</TITLE>
    <IMAGE>...</IMAGE>
    <ACTION>
        <TITLE>MIRROR LIGHT</TITLE>
        <SYSTEM>TELESWITCH</SYSTEM>
        <FEEDBACK><ADDR>1200</ADDR></FEEDBACK>
        <COMMAND>
            <TYPE>ON</TYPE>
            <EXECUTE>
                <![CDATA[swt?c=A&p=12862&a=2]]>
            </EXECUTE>
            <FEEDBACK IOMASK="0x40" STMASK="0x40"></FEEDBACK>
        </COMMAND>
        <COMMAND>...</COMMAND>
        <COMMAND>...</COMMAND>
    </ACTION>
    <ACTION>...</ACTION>
    <POSITION>...POSITION>
    <COORDS>...</COORDS>
</SPACE>
...
```

The action and command parameters are specific of the Teleswitch control and feedback system and therefore the <SYSTEM> tag should appear on the top of the <ACTION> definition.

Although the XML file was manually created, this type of configuration file could be generated automatically by a configuration tool provided by the system.

6.4.2 UDOMUS – User Interface (control and feedback console)

The user interface (control and feedback) is achieved by a software application in combination with a touch screen. This application was created to be a universal user interface that allows the interaction of the user with home automation systems, resulting in the name UDOMUS (Universal Domotics).

The application is basically a specific internet browser, based on the Microsoft internet explorer that uses html external files to define its specific interface. It also allows the access to external web interfaces to be displayed, which easily makes the integration in the console of external systems that supply web interfaces. An example of the possible integrations is the use of IP cameras or some alarm web communicators like JA-60WEB[1].

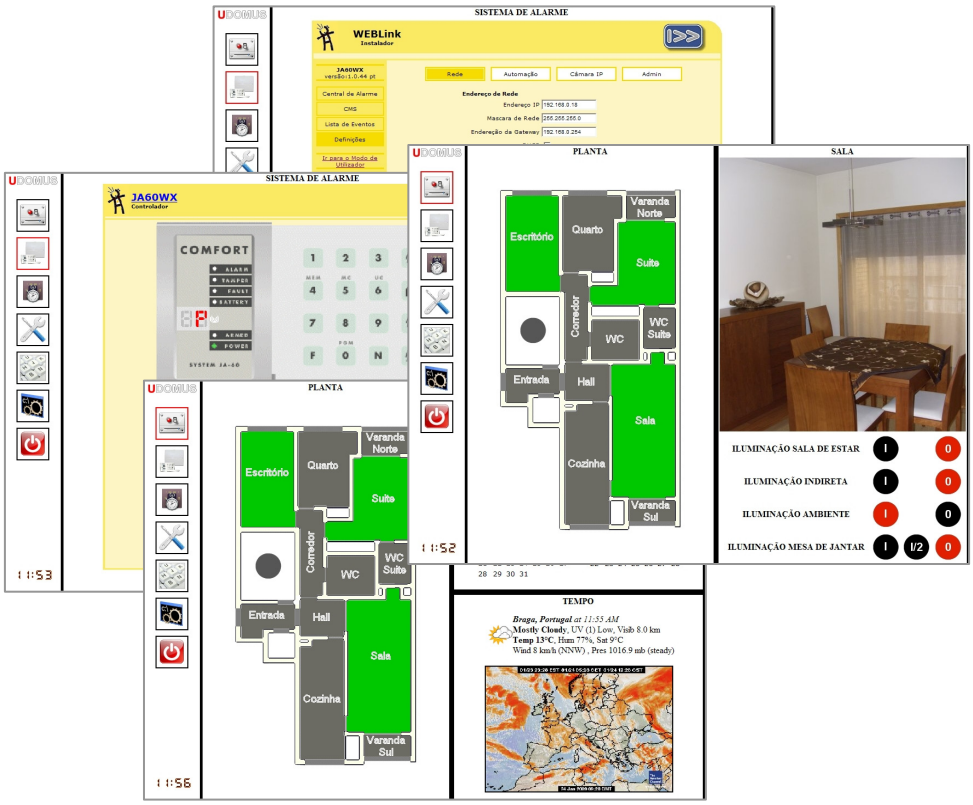


Figure 24 – UDOMUS application – screenshots

[1] JA-60WEB – Communicator module for the web interface to the commercial Jablotron alarm consoles models JA-63 e JA-65. More available on-line at: http://www.eurox10.com/Product/Security/JablotronSerie60/JA-60WEB.htm

In Figure 24 some screenshots of the console software are presented where it is possible to observe the some flat compartments highlighted in green, which indicate that some lights are turned on there. By touching the compartment, it is possible to see what lights are activated and the user may change its state, if needed. The state of the lights updates automatically immediately after they change state. Figure 24 also shows the access to the web base alarm system present in the flat. Other functionalities available on the internet can be added to the console software, as for example: weather forecast, map browsing, news services or even online shopping.

The application was programmed with c++ (visual studio 2003), using an interface with all the capabilities of the standard browsers, i.e., the interface can be defined in html, xml, JavaScript, Flash, ActiveX, etc. In addition, the application also includes some features, like the ability to use DLLs and a running application from the local html files. These are not included in standard browsers because of security reasons. The console also uses new and not standard html tags and definitions as presented in Table 12.

Table 12 – UDOMUS Meta Tag Options

UDOMUS Meta Tag	Description
<META name="UDOMUS" content="output-feedback">	Enables outputs feedback to be sent to the page via JavaScript Callback function.
<META name="UDOMUS" content="auto-shutdown:30">	Defines the no activity interval (in minutes) for console to auto-shutdown.
<META name="UDOMUS" content="no-activity">	Indicates that the page do not reset the auto-shutdown timer.

The html links in the console software also have a special format that allows to execute applications, redirect the output to a specific window or executing a http request, ignoring the output. To use these extra functionalities, it is only needed to add an extra string to the standard link, starting with *@@UDOMUS@,* as follows:

- *@@UDOMUS@RUN:[application path and name]@*
- *<standard link url>@@UDOMUS@TARGET:[main|none]@*

6.4.3 PDA UDOMUS – User Control and Feedback in a PDA

An application for Windows Mobile (WM5) was implemented in order to allow users to interact with the system anywhere in the house. In Figure 25, some screenshots of this application are presented. It uses the same XML description file as in the UDOMUS console application and specifically written html files for the PDA screen and capabilities.

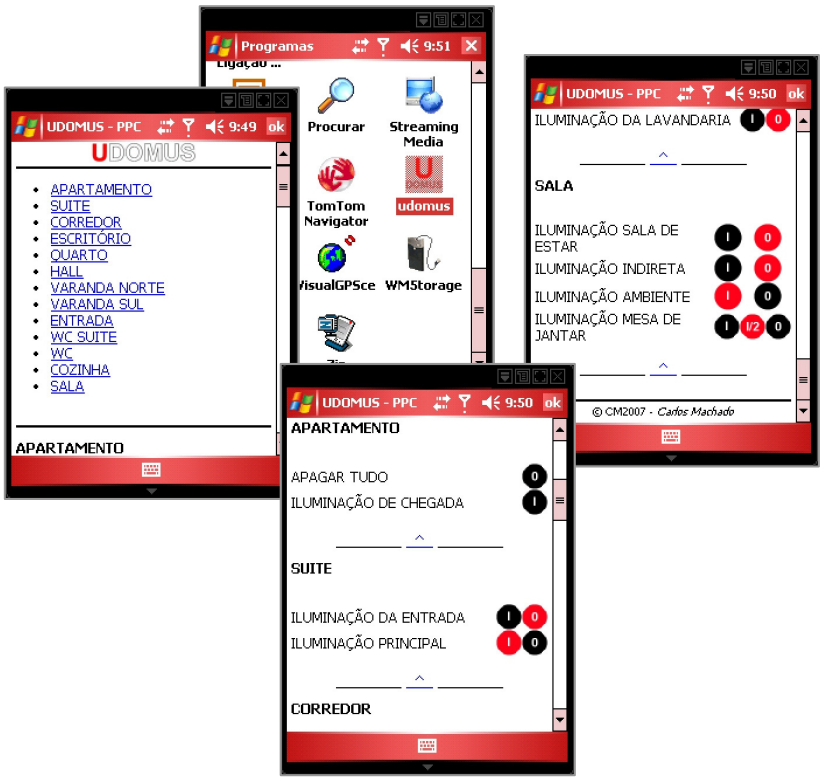


Figure 25 – WM5 version of UDOMUS application – screenshots

6.5 Summary and Conclusions

The case study installation in a real home and the constant usage by its inhabitants proved to be possible, without creating disruptions in their daily life. This was possible because the installation was performed step by step in an incremental way. The incremental installation was only possible because of the system philosophy, presented in this thesis, where the basic functionality existing in the environment devices does not depend on higher-level components and can work in stand-alone mode.

The robustness is another positive point in the case study implementation because the electrical light system was able to perform the basic functionality, without any failure. The capture of events has suffered some minor down times because the devices that support the Ethernet network (router, switch, etc) stopped working (freeze) and demanded resets, from time to time. That situation was not reflected in the basic functions of the flat and was only detected by a regular monitoring of the system.

The user interface system is placed in the hall, which is very centralized, where users pass by very often, proving to be useful for detection of forgotten turned on lights in the flat. That was possible because of the initial console screen that shows an overview of the apartment with the activated lights in the rooms. That situation has definitely changed the users´ behaviour in the flat mainly when leaving the house, where they inevitably look to the console screen to check if some light is left on.

The capture of the user interactions in the light switches started in 2007, July 1st and has continued for two consecutive years, having captured more than four million events. The down time of the capture system of some hours in this two year-time period was not noticeable. The analysis of the huge amount of data available includes the user activity patterns that can be recognized, unfolding the potential benefits of the ubiquitous system philosophy.

7 Building the context–awareness

The creation of the context-awareness is discussed in this chapter. The information gathered from the sensors and actuators is processed in order to retrieve the context, predict the routines and allow the execution of an autonomic action adaptable to interactions from the users. The context construction uses data mining techniques going from basic data statistics to more complex AI neural networks techniques. The output usages, as well as patterns from user behaviour are analyzed in order to make system autonomic actions possible.

7.1 Context-awareness

The context-awareness is a crucial characteristic in the construction of ubiquitous systems. Only with the context-awareness it is possible to create applications that react to users behaviour, i.e., all the devices in the environment need to be aware of the context, in which user actions are done, in order to auto tuning to that particular situation.

The construction of the context-awareness in an environment where devices are abundant must be adaptable and autonomic, i.e., the system needs to be capable to learn behaviour and automatically adjust its actions in function of the context.

The requirement of systems being able to learn context suggests the usage of artificial intelligence (AI) algorithms as a basis to process the raw data and acquire the context from it. However, not all AI algorithms can be used to determine the context because some require constant feedback from users. That feedback is impractical in environments full of devices resulting in constant user input and consequently colliding with Weiser's ubiquitous system view. In other words, the usage of AI algorithms that require supervised-learning should be avoided in the implementation of ubiquitous system context-awareness computation.

Another important aspect of the AI algorithms is the need to a permanent adaptation, using the acquired data, in order to adjust to real environments where routines can change according to new circumstances, i.e., if a user changes the working schedule, the routines related with leaving and arriving time home can change, and consequently the system must adapt to this new situation. This characteristic suggests that the system should constantly capture and store the data from the devices, in order to execute AI learning algorithms.

The implementation of AI algorithms should ideally be distributed but that is impracticable in low cost devices with very little resource. The solution to this problem is to use a centralized device with the necessary resources to execute the learning algorithms for the AI. To try to minimizing the risk of failure of the centralized device, the AI algorithms to be used must be able to cope with missing data in the learning process.

After the completion of the training process, the AI algorithms that require low resources are preferable, because the knowledge gained can be distributed throughout the environment devices. This concept allows the centralized device where the AI learning is executed, to be off-line during relatively small intervals without affecting the intelligence of the system. The learning process can even be delayed to idle hours, only requiring the capture and register data procedure, allowing the centralized device to execute other tasks like visualizations or users interface/feedback.

7.2 Recognition of Daily Life Activities

Smart environments are being developed, by academia and industry, around the world, e.g., by AwareHome [Kidd et al., 1999], intending to create a living laboratory for ubiquitous computing research on everyday activities; the MavHome project [Cook et al. 2003] is a multi-disciplinary research focused on the creation of an intelligent home environment from University of Texas at Arlington.

From industry, the Philips Company established HomeLab[1] as a testing ground advanced interaction technology. This collaborative product and technology development, PlaceLab project [Intille et al., 2004], is a joint initiative between Massachusetts Institute of Technology (MIT) and TIAX firm. This provides a living laboratory to study human behaviour, their routine activities and interactions of everyday life.

The recognition of Activities of Daily Life (ADL), is addressed by almost all researches in order to achieve high context definitions to be used by other high level applications.

In recent years there has been a growing usage of diverse data mining techniques to process captured data from a collection of sensors scattered in the environment, aiming the calculation of patterns of the human activity, e.g., in [Kautz et al. 2003] the usage of a hierarchical hidden semi-Markov model to track the daily activities of residents in an assisted living community; Tapia [Tapia et al. 2004b] applied Naive Bayesian classifier to recognize ADL such as bathing, toileting, dressing and preparing lunch, based on the analysis of data collected from a set of small and simple state-change sensors; the usage of a temporal neural network-based agent [Rivera-Illingworth et al. 2006] to recognize behaviour and ADL such as listening to music, working on the computer and sleeping; it utilizes a user interface (UI) to help to label activities of the users; [Lühr et al. 2007] uses a data mining approach, an intertransaction association rule (IAR), for the detection of new and changing behaviour in people living in a smart home; [Zheng et al. 2008] used a self-adaptive neural network (SANN) called Growing Self-Organizing Maps (GSOM), a cluster analysis of human activities of daily living.

The techniques mentioned are lacking one or more particular aspects that invalidate their use in a real implementation or do not reflect the ubiquitous computing philosophy. Some require the users input to label the states, others are too complex to be distributed throughout the environment devices and others do not work in a fully automated process. The method presented in the next section, and published in the paper [Machado et al. 2008], tries to address the aspects of processing data in a complete autonomous procedure, using simple statistics and a simple

[1] Philips Research– HomeLab. URL: http://www.research.philips.com/technologies/projects/homelab/index.html

structure of Artificial Neural Networks (ANN), to learn the behaviour in a configuration where the knowledge can be inserted in the environment devices.

7.3 Case Study: Automatic Light Control using ANN

The case study that is described bellow shows how it is possible to achieve self-configuration of the light control in a real home scenario, using the guide lines discussed earlier, i.e., the ubiquitous computing system vision, where the patterns of the actions of the users are captured and recorded to create the context-awareness. The system tries to free users from setup adjustments as the home tries to adapt to its inhabitants' real habits.

Artificial Neural Network (ANN) is used as a pattern recognition method, classifying for each moment the light state. The data used to train, test and evaluate was acquired from a real house where a family is actually living, as described in the previous chapter.

The ANN algorithm, versus other A.I. technique, like knearest neighbour algorithm (kNN) was used because it can reduce an megabyte sized events database to a few Kilobytes arranged as ANN weights. This compact information is ideal, after training, to distribute it throughout the scattered embedded devices in the environment, with limited resources.

7.3.1 Data Acquisition

Data acquisition is the first step of a pattern recognition system as described by Polikar [Polikar, 2006]. The data was collected from a real instrumented home where a young family, composed by a couple and a child, is actually living.

In the implementation, it is important that the acquisition of the users actions is done in an implicit way, i.e., in such a manner that is completely effortless for the inhabitants of the home [Schmidt, 1999]. This aspect was considered in development of the Teleswitch device, which

controls the instrumented house. This device was designed to work in a distributed network and taking into account some characteristics that allow it to be included in an autonomic ubiquitous system as described in the paper published [Machado et al. 2007].

The events are captured by the Teleswitch device, and then they are transmitted to a communicator device (the gateway of the dedicated home network) where they are time stamped. Finally, the events are stored in a database. Table 13 shows an example of the events that report the actions of the users and the actuator actions. Typically, to every input event, an output event is generated. The first event is the input caused by users and the second event is the device resulting action.

Table 13 – Events sample when switching a light bulb

Event Time Stamp	Device Address	IO Identification	IO Type	Action
18-03-2008 6:23	18	3	Input	Key Pressed
18-03-2008 6:23	16	130	Output	Relay On
18-03-2008 6:39	18	3	Input	Key Released
18-03-2008 6:39	16	130	Output	Relay Off

The events in the database can be used to calculate several statistics, as a training-set to machine learning or in data mining algorithms. In Figure 26 it is presented the light accumulated usage charts of four spaces selected from the instrumented house. In the kitchen light usage, it is possible to see that the dinner is served around 8 p.m. (on average) and after that, the family goes to the living-room. The hall light also indicates that its usage increases around 7 p.m., possibly because that is the home arriving hour.

Another accumulated value presented in the kitchen, WC mirror and hall around 6 a.m. to 7 a.m. is very noticeable. It indicates that a very precise and timed event occurs almost every day. This feature can be easily identified by the get-up and go-to-work routine. The accumulated value in

the WC mirror light starts at 6 a.m., in the kitchen and in the Hall it starts about 6:30 a.m. and ends around 7 a.m..

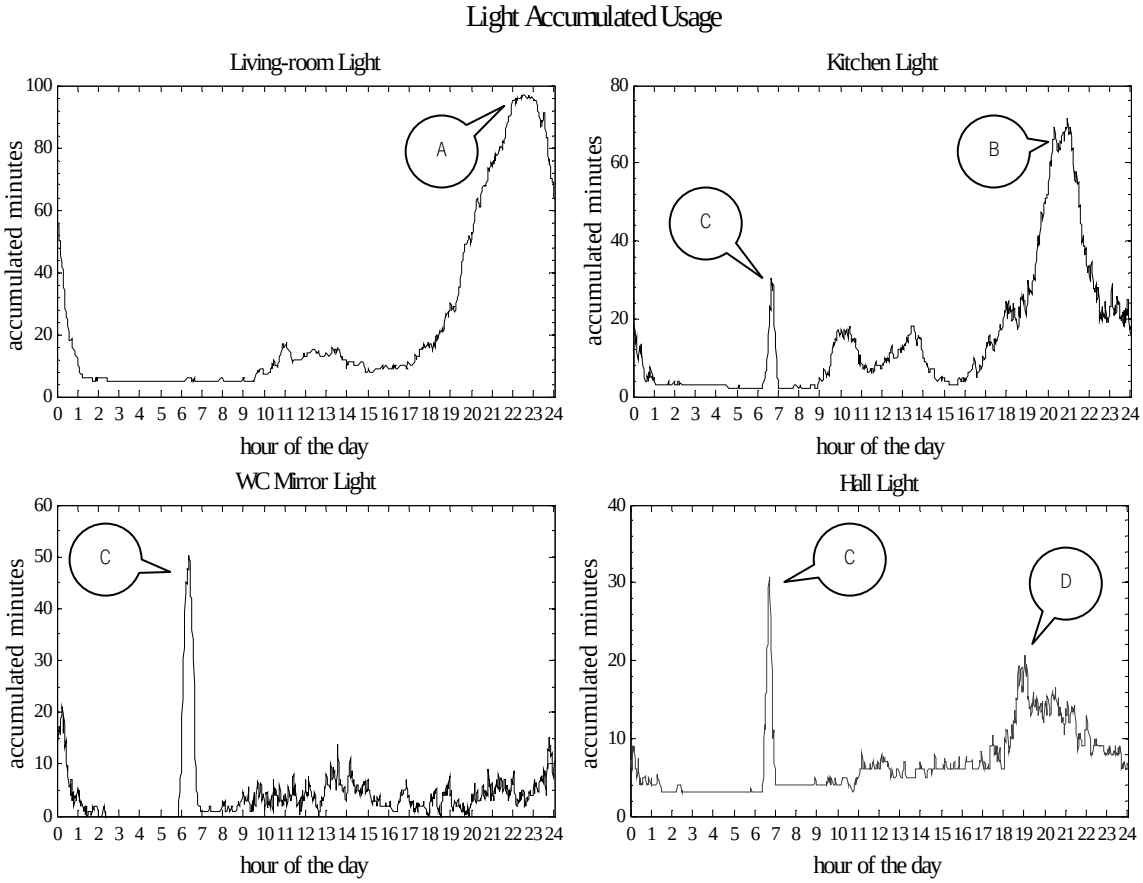


Figure 26 – Light accumulated usage with minute resolution (122 days period); Labels: A- Living-Room (watching tv); B- Dinner; C- Get-up and go to work; D- Arriving home from work;

The histograms shown below in

Figure 27 point out the usage intervals in which a light is switched-on by the inhabitants. These histograms are very important to determine the moment it is probable to switch-off a light. It is possible to see that 90% of the usage intervals for the WC mirror light has a duration of less that 1500s (25 minutes) and almost near 100% of the activities happen in less than 3600s (1 hour). In the hall light usage, the percentile of 90% happens after an hour and the value near 100% only

appears later on. This indicates that an automatic decision to turn-off the light needs to take into account patterns from users to prevent a wrong action. The light usage intervals histogram values below 30s were filtered because they have less impact in decision making and in light power consumption.

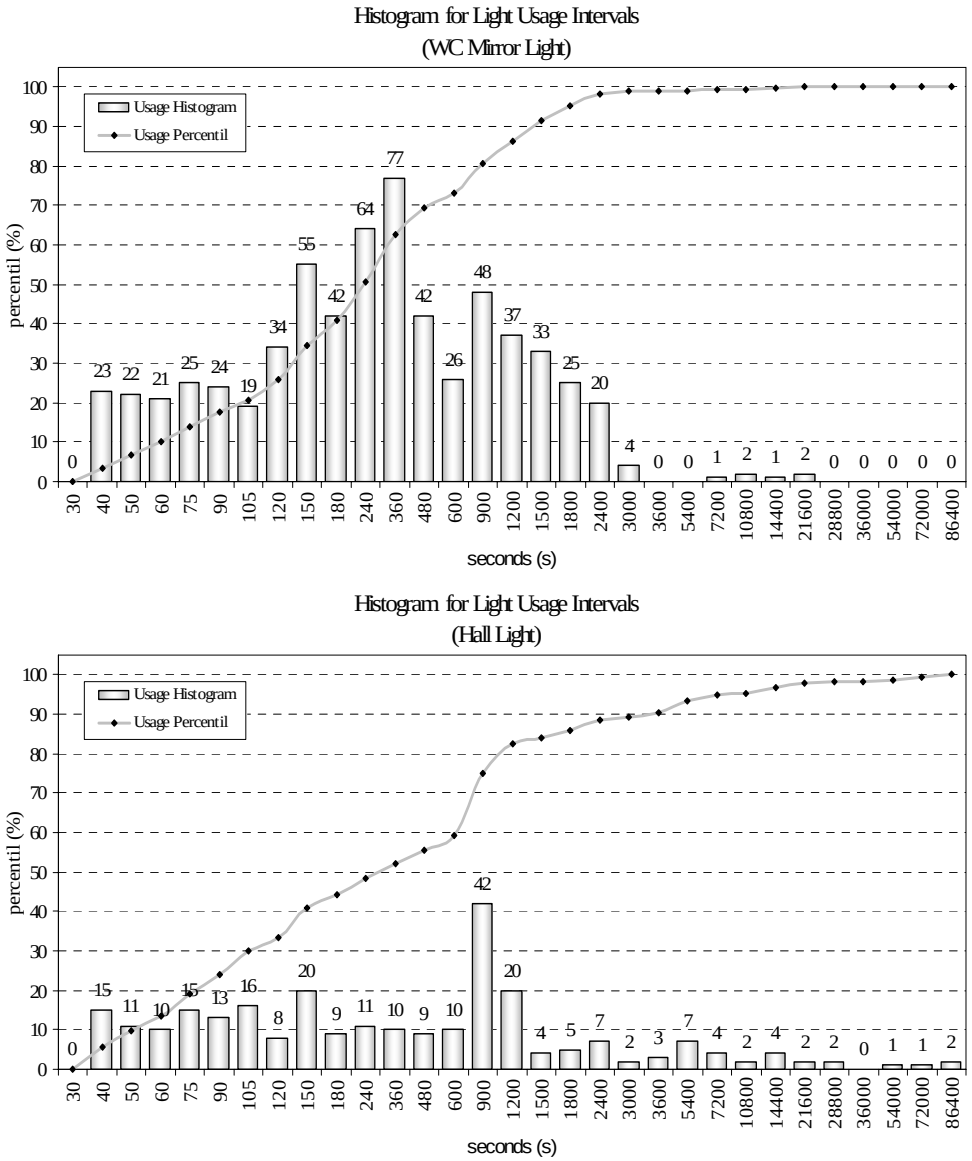


Figure 27 – Histograms for light usage intervals (122 days period)

7.3.2 Data Set

The data captured indicates that the light usages are somehow related and follow a pattern. This means that it may be possible to predict the state of light bulb by looking back to the previous events. It is also necessary to take into account the hour of the day and the weekday to improve the prediction. The weekday helps by separating patterns by users in a normal working day from that of the weekend.

The dataset needs to be prepared before it can be used to train the ANN. The objective of the ANN is to classify the state of the light in order to predict and control it. As input vectors, it necessary to supply the ANN with cases when the light state is ON and cases when it is OFF.

Pre-Processing

The first step is creating a record set to every light source where, for each moment, identify the light state and time of the state change, i.e. a record set of tokens of {light state (s); time passed from the last occurrence (t)}. The following step is filling in the blanks by expanding the data to get a record for each minute of the day for the time interval selected.

A ratio between records can be determined, indicating the symmetry of the training data. An unsymmetrical data indicate that more records are needed to train ANN successfully. In this case, the ratio depends on the amount of time a light is used in the data interval selected for processing. In the implementation was obtained a light ON state ratio of 1:49 for the hall light, but a better ratio 1:6 for the living-room light.

Finally, and before the normalization procedure, the dataset needs to be converted from the token into a single value. The equation 1 presented bellow does the transformation, where τ indicates the maximum time an event considered.

$$v(s,t) = \begin{cases} (\tau - t) \times (2s - 1) & (\tau - t) \geq 0 \\ 0 & (\tau - t) < 0 \end{cases} \quad (1)$$

The resulting dataset of the equation has fading values in the interval [0..τ] for lights that were deactivated and [-τ..0] for lights that were activated in the last τ time. Using this equation, it also prevents that the light left ON does not disturb the patterns detection that may happen later on. In the implementation, it was selected a equal to 60 minutes, which means that, in this case-study, only events that happened in the last 60 minutes are taken into account.

Normalization

The normalization process is very important because it speeds up the training phase [Shalabi et al. 2006]. There are many methods for data normalization: min-max normalization, z-score normalization and normalization by decimal scaling. In this implementation the min-max normalization has been selected, because the minimum and maximum are well-known. Min-max normalization performs a linear transformation on the original data. If it is considered that min_a and max_a are the minimum and the maximum values for attribute A, the min-max normalization maps a value *v* of *A* to *v'* in the range [*j, k*] by computing the equation 2.

$$v'(v) = \frac{v - \min_a}{\max_a - \min_a} \times (k - j) + j \quad j < k \quad (2)$$

In the dataset the values in the interval [-τ..τ] are transformed into the interval [-1..1], simplifying the min-max calculation and resulting in the equation 3.

$$v'(v) = \frac{v}{\tau} \quad (3)$$

The time interval (minutes of the day) was also normalized using min-max normalization, converting linearly the interval [0..1439] to values within [0..1] and the same normalization was done in the weekday attribute.

7.3.3 Artificial Neural Network

In order to obtain automatic classification of light state, it was applied artificial neural networks (with feed-forward full connectionist structure and with sigmoid activation rule, as described by Gurney [Gurney, 1997]) in the lights usage dataset described in the previous section.

Every input vector contains the time, the day of the week and the last event time for each light in the house, except the light that is being predicted. The value of the light state to be predicted is used to classify the datasets.

ANN Structure

The ANN used is composed by three layers (input layer, hidden layer and output layer). The input layer has 26 neurons (one for each of the 24 light sources and one input neuron for the time attribute and another for the weekday attribute). The hidden layer uses the same number of neurons as the input layer. The numbers of the hidden layer neurons can not be too small, because it decreases the capability for the ANN learn patterns, but if the number of the hidden neuron is too large, there is a risk of over-fitting the dataset [Michael et al., 2004]. The output layer is composed by a single neuron.

The output *y* of the single ANN layer is calculated in the equation 4. The *W* represents the matrix layer neurons weights, the *x* is the input vector, *b* is the neurons bias weights, and *f()* is the activation function, which is the sigmoid function shown in equation 5.

$$y = f(Wx + b) \tag{4}$$

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{5}$$

ANN Training

The backpropagation algorithm with momentum was used for ANN classifier training. This algorithm is a common method of teaching ANNs, and it is most useful for feed-forward networks (networks that have no feedback, i.e., that have no loop connections). On the algorithm iteration, the single layer matrix *W* update is defined in equation 6 where α is the momentum, η is the learning rule and δ is the neuron error.

$$\Delta W_{ij}(t) = (1 - \alpha)\eta x \delta + \alpha \Delta W_{ij}(t - 1) \tag{6}$$

As a way of evaluating the binary classification, it has used the Sensitivity (*Se*) and the Specificity (*Sp*) statistical measurements.

The Sensitivity or recall rate specifies the portion of true positives (*TP*) in the classification, i.e., the portion of the light ON state that was positively detected. The *Se* is calculated using the equation 7, where *TP* (true positives) is the number of positive cases correctly classified and *FN* (false negatives) is the number of misclassifications for positive cases being incorrectly classified as negative.

$$Se = \frac{TP}{(TP + FN)} \tag{7}$$

The Specificity is the proportion of true negatives (*TN*) in all negative cases in the dataset, i.e., the fraction of the light OFF state that was correctly detected. The *Sp* is determined by calculating equation 8, where *TN* (true negative) is the number of correct classification of the "light OFF" state and *FP* (false positive) is the number of misclassifications where it was incorrectly classified as a "ON state" of the light.

$$Sp = \frac{TN}{(TN + FP)} \tag{8}$$

In the classification of the light "ON state", the *Sp* needs to be as high as possible because a *FP* (false positive) is unacceptable in a home environment since it activates a light when it is not the right time to do that. A low value of *Se* is not so critical because it is possible to cross-check with light usage intervals histogram before executing an action.

7.3.4 Evaluation and Results

To evaluate the ANN classifier, the hall light of the instrumented home was selected, in order to verify how well the light state is predicted. In this particular light, and for the interval analyzed, a total of 178028 records here obtained, from which the 3592 records are ON state and 174436 are OFF state (ratio of 1:49). The ANN the momentum (α) is 0.7 and the learning rule (η) is 0.2.

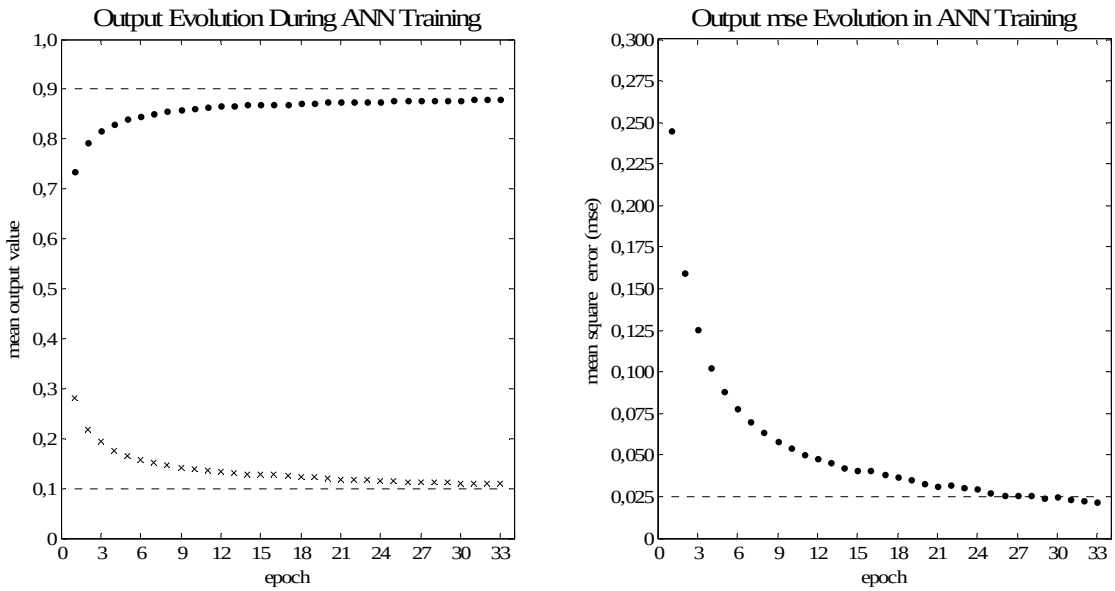


Figure 28 – ANN epoch evolution during training

The ANN took 13 minutes to train in a 1.5 Ghz cpu and converged in 33 epoch for a mean square error (*mse*) threshold of 0,025. The training stopped after a 5 consecutive epoch below the *mse* threshold as shown in Figure 28.

Table 14 – Statistical measurements from the ANN classification of the hall light state

	Sensitivity *Se*	Specificity *Sp*	Accuracy *Ac*
Training Set	93.95 %	96.78 %	96.72 %
Validation Set	84.63 %	96.64 %	96.40 %
Test Set	85.88 %	96.46 %	96.25 %

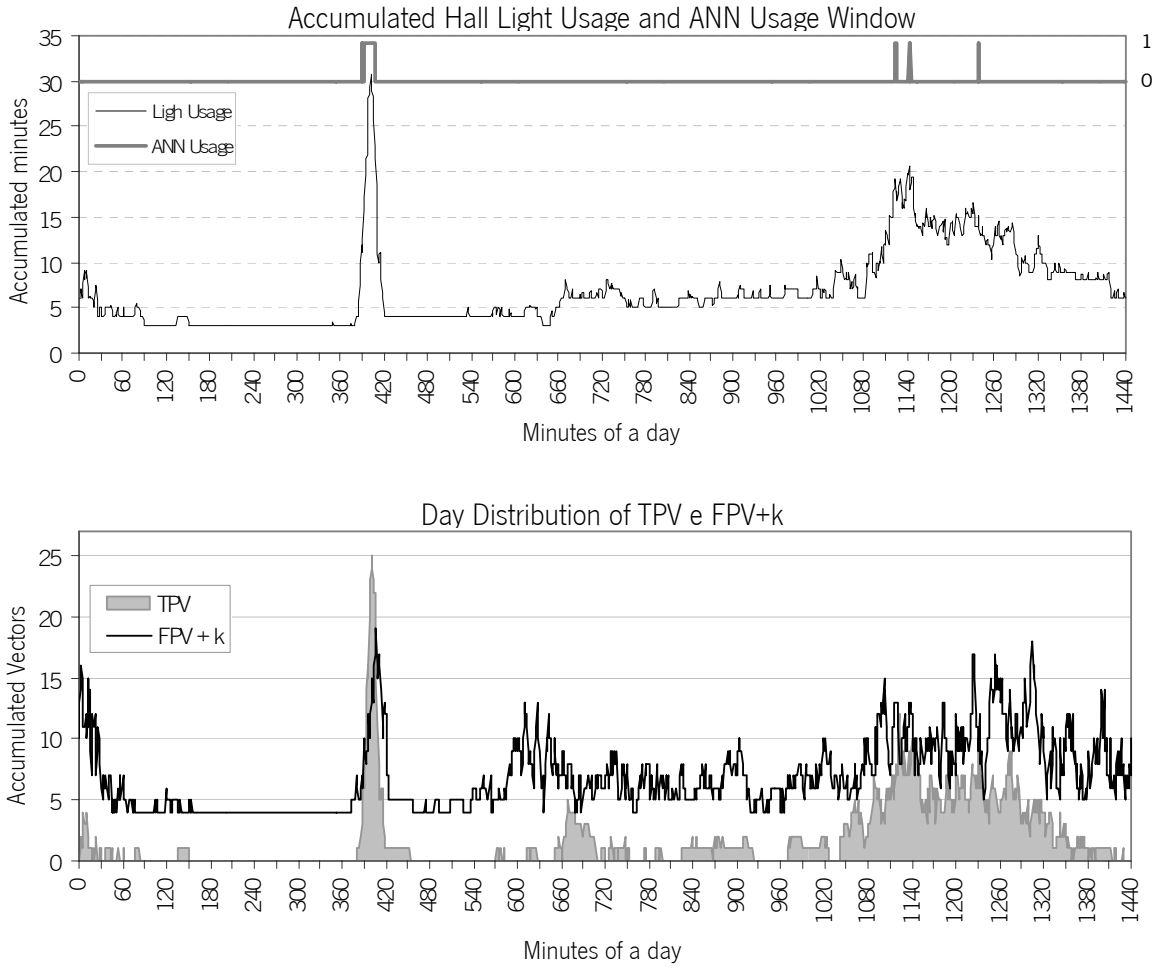


Figure 29 – Daily Distribution of True Positive Vectors (TPV), False Positive Vectors (FPV), Accumulated Hall Light Usage and ANN usage window ($\kappa = 4$)

In Table 14, the *Se* indicates that the ANN can predict more that 80% of the day real light "ON states", which means that these states are somehow related with other previous actions and that the pattern has been learned by the ANN.

The Accuracy (Ac) and the *Sp* show a large value (96%) indicating that the ANN is working well, but because of the large number of the light OFF states records, the absolute number of false positives (FP) is still very high. In Figure 29 it can be seen that the FPV (False Positives Vectors) appear almost everywhere the activity was reported.

One of the characteristics of the ANN is the ability to generalize and that can explain some of the errors in the classification because the data is based on the users patterns that do not always

choose to turn on the light as they are used to. But the ANN has classification errors that are very important and there is a need to recognize where that is happening.

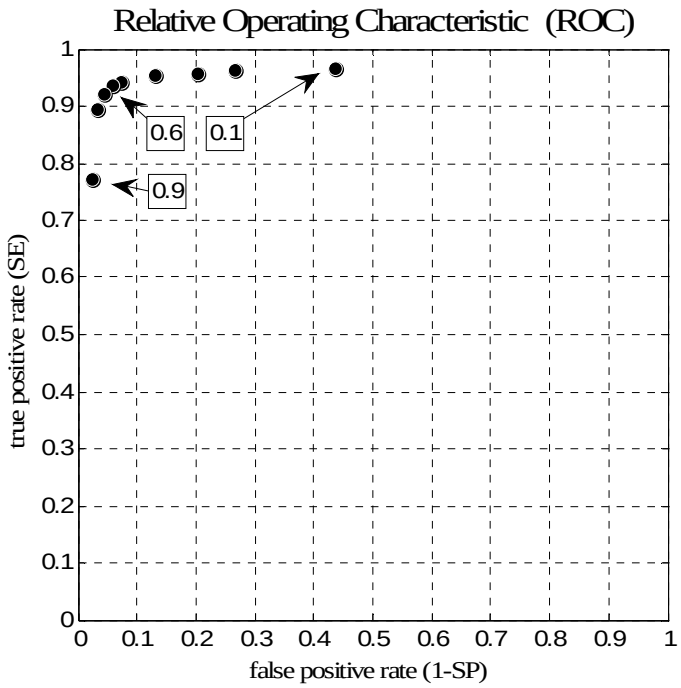


Figure 30 – Relative Operation Characteristic (ROC) for ANN threshold

Figure 30 shows the relative operation characteristic (ROC[1]) [Beck et al., 1986] for the ANN classifier, i.e., the classification evolution in function of the output threshold. It is possible to observe that the threshold of 0.6 increases the results of the binary classification of both *SE* and *SP*. But in this particular application, a better compromise is the threshold of 0.8, because the *SP* is greater and *SE* is still around 0.9.

[1] ROC – Relative operating characteristic, also known as Receiver Operating Characteristic, are graphs that organize classifiers by visualizing their performance. ROC graphs are commonly used in medical decision making, and in recent years have been used increasingly in machine learning and data mining research. More information available on-line at: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

Activating the light

In home automation scenario, the wrong actions from the system should be minimized. The usage of the ANN classification should be used only in the periods of the day that show a better performance. The performance indicator (P_m) of the classification can be calculated by the equation 9, where the TPV_m and FPV_m is the True Positive and False Positive Vectors respectively for a specific minute of the day.

$$\mathrm{P}_m = TPV_m - \left(FPV_m + \kappa\right) \tag{9}$$

A positive performance value indicates that there are more TPV_m than FPV_m and the ANN can be used on that minute. The constant κ introduces a bias to the performance indicator, allowing the setup of the performance level.

In this case-study, with κ=4, the ANN has the usage windows shown in Figure 29. The ANN usage windows appear at the same time as the higher Hall Light accumulated usage, but in the minute 8:38 p.m. it is not that case. This may happen because it could be a new pattern or a pattern which only happens on some specific days.

In order to improve the processing and to limit the system actions, the call to the ANN classifier can be constrained to the moments where a user event occurs. This guarantees that an action is always the result of an event that is caused by a user.

Deactivating the light

To deactivate the light, the utilization of the ANN should be complemented using the normalized accumulated usage *a()* and histogram percentile (

Figure 27) *p()*. Equation 10 shows how to calculate a value of the function *D()*, which fuses the information of the light usage statistics. In case it is above a threshold, it points out the moment on which the ANN classifier can be invoked. This function takes into account the duration of the

usage $p(t-t_0)$, the quantity of the usage for a particular time $a(t)$ and the progress of the accumulated usage $a'(t)$.

$$D(t) = p(t - t_0) - a(t) - \varphi.a'(t), \qquad \varphi \geq 0 \tag{10}$$

The usage of the ANN classifier for deactivating the light allows the decision threshold to be lower, resulting in higher power savings, as it may turn off the light sooner, as it takes into account the habits of the users.

7.3.5 Spreading the behaviour into the environment

The process of training the ANN needs to be frequently executed in order to incorporate new patterns and disable older ones. The training process is executed everyday, by the house computer, at night hours, has a low priority process. The training process uses a window (122 days in our implementation) with the latest event data. The frequency of the training is sufficient to maintain the system updated.

After the training of ANN, for each light source, the information is sent to the embedded devices in the environment. These devices monitor the events related with usage of the lights and execute the ANN classification, acting in the environment as already described. In this manner, the system has a lower latency and, at same time, a more robust system can be obtained, i.e., if the house computer is down for some reason (blocked, rebooting or temporarily power-down) the environment automatic actions still work. Any missing data caused by the failure of the home computer do not have a great impact on the system automatic action, because of the usage of ANN and basic statistical calculation, i.e., if a computer fails one day in an interval of 122 days, less than 1% of data is missing.

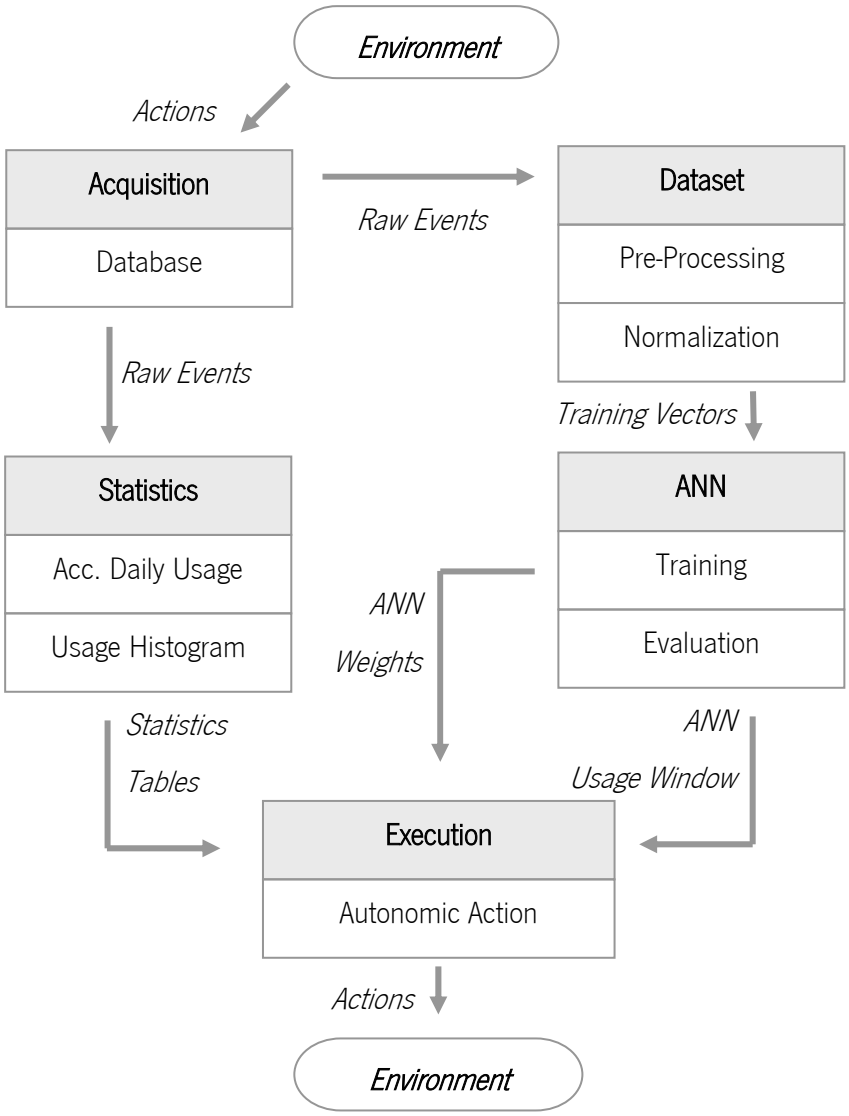


Figure 31 – Autonomic Light Control - Process Diagram

The process diagram presented in Figure 31 shows the steps in data processing from the raw events to the learning behaviour of the lights usage. The processing of events, for this case-study, resulted in three data collections: the statistics tables that incorporate and emphasize the normal handling of the lights; the ANN weights incorporate the relation between the light state, the previous actions and users behaviour; the ANN usage window defines the moments where the system uses the behaviour learned.

The resulting data collections are relatively small which allows them to be deployed to the environment devices. These only need to intercept the events in the data bus and execute the automatic light control algorithm.

7.4 Summary and Conclusions

In this chapter it is discussed the construction of the system context-awareness in order to execute autonomic actions in a reflexive way, without the users explicit intervention, but reflecting their habits.

Many AI algorithms can be used to process the available event data supplied by the sensor and actuators in order to distinguish patterns from users and learn there behaviour; but not all meet the requirements for the construction of a ubiquitous system. Some AI algorithms require direct feedback from the users, some involve a supervised learning step and others require large resources available (large storage or processing power).

The distributed nature of the ubiquitous systems implies the usage of AI algorithms that can work in a distributed way. Sometimes this requirement is impossible to achieve due to various limitations. When that happens, an alternative and good solution is to embed the knowledge learning into the devices scattered in the environment, separating the training step from the execution step of the AI algorithm. This can be used whenever patterns from users change slowly over time, which is a typical situation in a home environment.

The robustness of the autonomic actions is another important aspect that needs to be addressed, i.e., in the environment where users normally have well-established routines, a wrong autonomic action from the system can be inopportune as well as a constant request for user confirmation. The system needs to have a high level of confidence before it takes the initiative of executing an action, considering that no action from the system is much better than a wrong action.

8 Results and Evaluations

The data captured by the system is analysed, in search of any patterns that may emerge during the normal daily activities of the users. The analysed data is composed by events captured over a period of two complete years, ranging from 1st July 2007 to 31st July 2009. This two year period allows the detection of a global and long range tendency in behaviour of the users as well as changes in the habits of the users during the above mentioned period.

The quality of the data captured in this project validates the selection of the system architecture used in this project, showing the ability to make a continuous and not obstructed capturing of the activities of the users a reality.

8.1 Captured Data Global Analysis

Figure 32 shows the accumulated events per day during this two year-study. The total number of captured events from the data bus in the study is 4149218. The events represent the direct and indirect interactions of the users with the system. Some events are caused by switches that have changed their state. Others events are related with the action of the actuators. The motion detectors are responsible for detecting movement activity in the house.

The number of accumulated events is greater when there is more activity in the house. The number of events is lower (or even zero) whenever the family gets out. As an example, seven dates were pointed out in Figure 32: the Christmas Eve (24-12-2007, 24-12-2008) zero or low number of events has captured; on the child´s birthday party (on 19-01-2008 and on 01-02-2009 - which was on a Sunday, eleven days after his birthday) were captured over ten thousand events; the woman´s birthday (on 18-11-2009) and the man´s (on 20-12-2007 and 21-12-2008 - which was on the Sunday after birthday). In the year 2007 no events were captured, whereas in the following year fourteen thousand events were captured, indicating a house party.

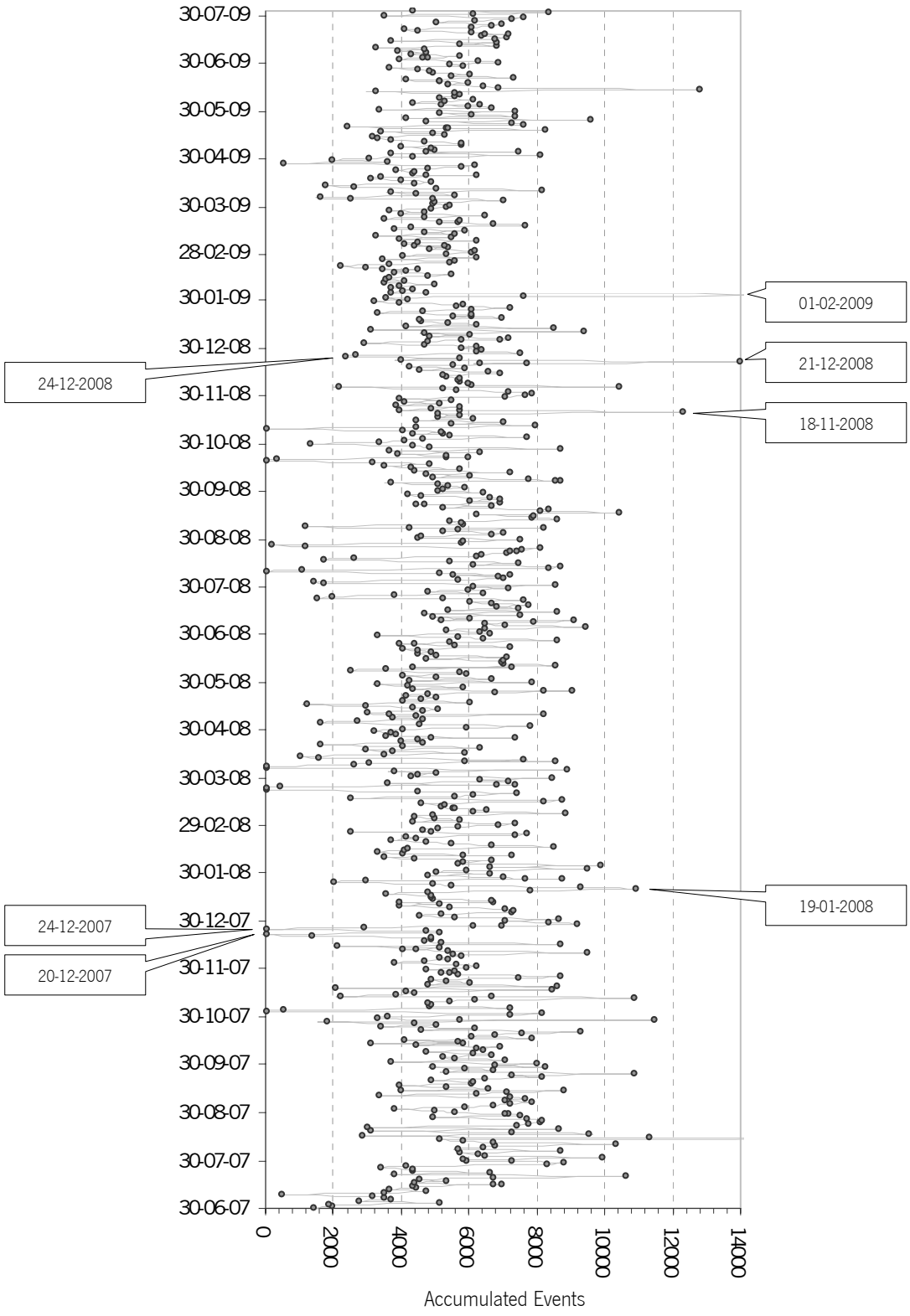


Figure 32 – Accumulated events per day (during 2 years)

The events capture per day have a calculated a mean value of 5431 events, with a standard deviation of 2158 events. These simple statistical measurements can be used to create a common-day-index which is useful to augment the confidence in pattern recognition based on a statistical measure algorithms.

The histogram presented in Figure 33 shows the distribution of the accumulated events per day. It shows that 44% of the day has accumulated events between 4000 and 7000. The figure also divides the classes into the 4 quarters of the years. This division allows detecting that the days from the 1st and the 2nd quarters of the year accumulate less events compared to those from the 3rd and 4th quarters. This indicates that the accumulated events per day change along the year, pointing out that the behaviour of the users change throughout time.

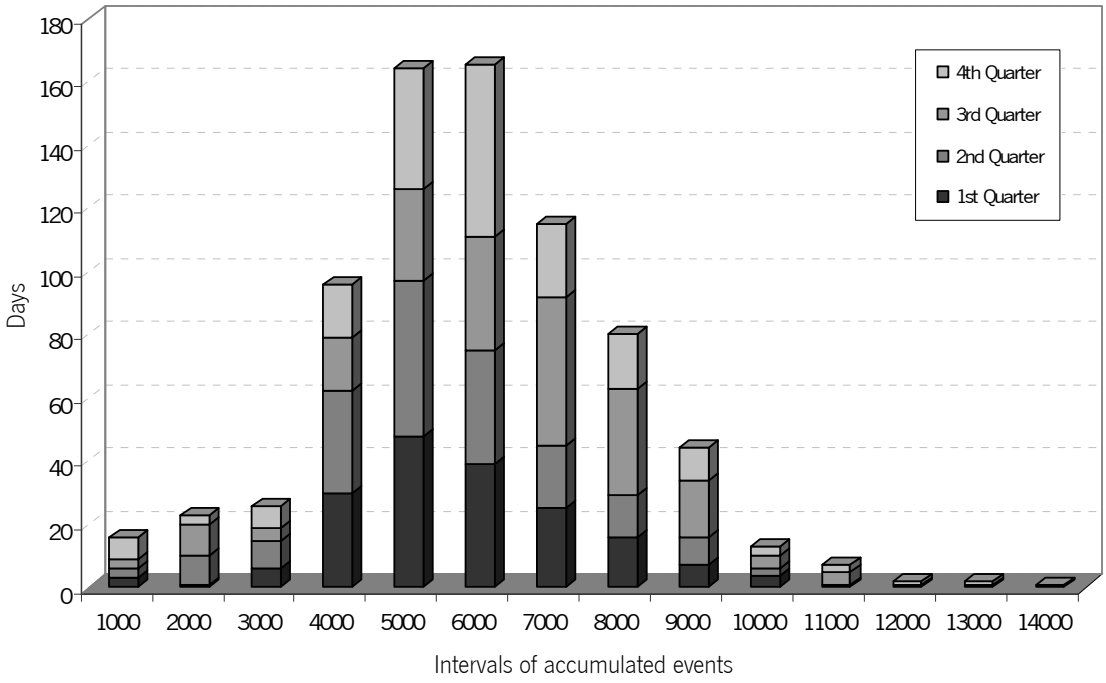


Figure 33 – Histogram distribution for the accumulated events/day (during 2 years)

8.2 Users Annual Behaviour

Humans are considered creatures of habits. This characteristic is present not only on the short term (daily behaviour) as well as in the long term (year behaviour). In a house, the activities of the users are also dependent on the weekday, because that defines, in general, whether the users are engaged in working activities, or the users have more freedom to do different and non-systematic tasks at the weekend.

In Figure 34 it is show the evolution of accumulated events per month and grouped by weekday. In order to allow a better perception of patterns, the chart surface was smoothed out using a Gaussian function. This representation allows the observation of some patterns on the First year which were repeated in the Second year.

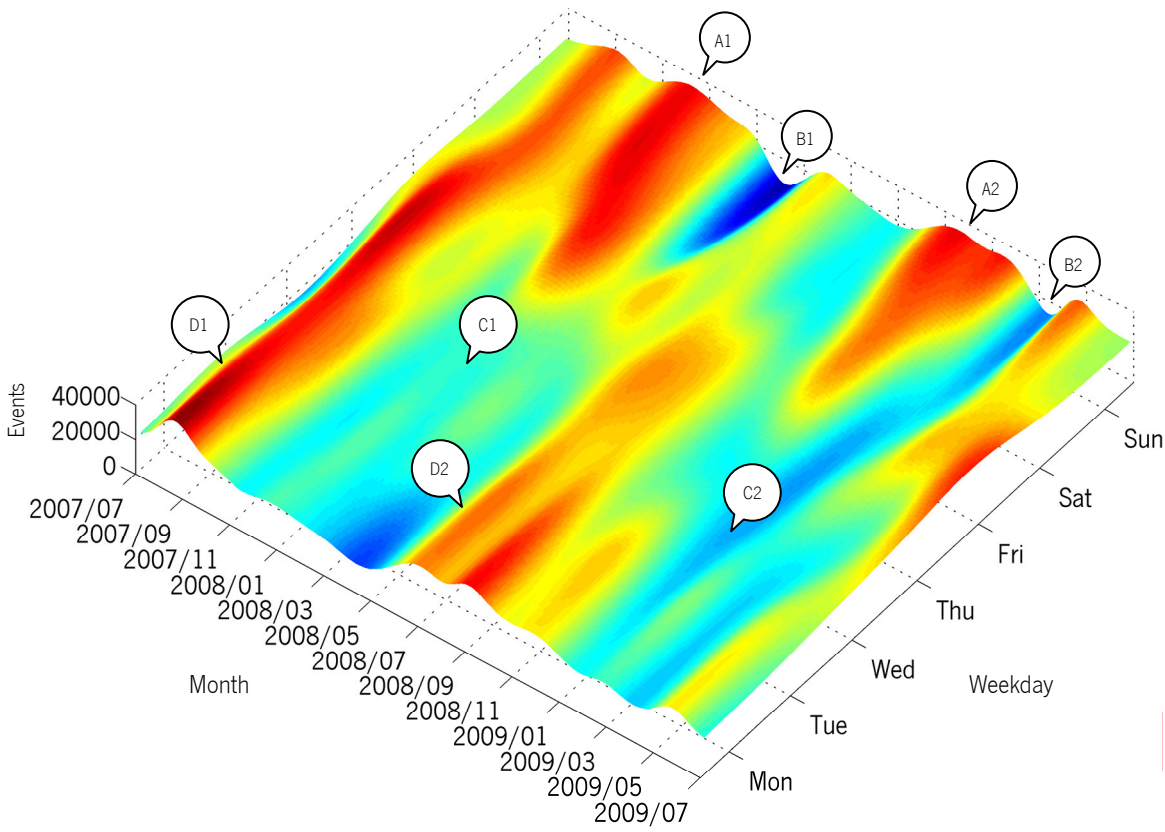


Figure 34 – Accumulated Events per Weekday and per Month (during 2 years), Labels: A- Christmas Holidays; B – Easter Holidays ; C- Working Days; D- Vacations

On the surface presented in Figure 34, it is possible to observe that the accumulated value of events is, in general, lower and more stable on working days (Mon, Tue, Wed, Thu and Fri). The value of accumulated events on working days tends to increase specially in August which can be explained by the family vacations.

Another feature that is noticeable on the surface presented is the peaks (higher and lower accumulated events at home) that appear on the weekend days (Sundays and Saturdays). The weekends that present higher accumulated events are those around December. That can be explained by the birthday parties and Christmas school vacations. In contrast, on Easter Holidays it is easily noticeable a reduction in the accumulated events in the home. This feature indicates that the family stays out for that particular time, according to the years studied. The year 2008 has even a deeper value because of particular events that have forced the family to stay out of the house during the weekend in order to assist family members.

All features identified in the first year data are in some degree repeated in the second year, showing that behaviour is cyclic. The information represented aggregates data from several days of each month and therefore it can be only used as an indicator to the probable activity in the home. This indicator can help an automatic heating/cooling system of a home to make the decision of predicting its activation/deactivation, i.e., making decisions according to context.

8.3 Daily Behaviour Profile of the Users

The human activity during a day shows a sequence of events that can be cleared observed in Figure 35. The captured data reflects the human behaviour that take place during a typical day at home. The weekday also may have an impact in daily behaviour profile of the users, because not all the users have lunch at home during working days.

In Figure 35, it is presented the accumulated number of daily events, grouped by the weekday, during the two-year study period. Again, in order to allow a better perception, the surface

was smooth out, using a Gaussian function. That shows the typical daily profile, where it is possible to observe that, on average, that day starts at 9am and ends around 1am (normally 8 hours of sleep). This time interval represents a total of 1/3 of the day where the system has little demand from the users and therefore can be used for execution of heavy and offline processing algorithms. As an example, there is the re-training phase of ANN, statistic calculations, system adjustments and maintenance.

Figure 35 also shows that the weekends have a different daily profile from that of the working days, e.g., during the weekends more events are accumulated rather than the week days. In general, the lunch is served around 1pm and dinner around 8pm.

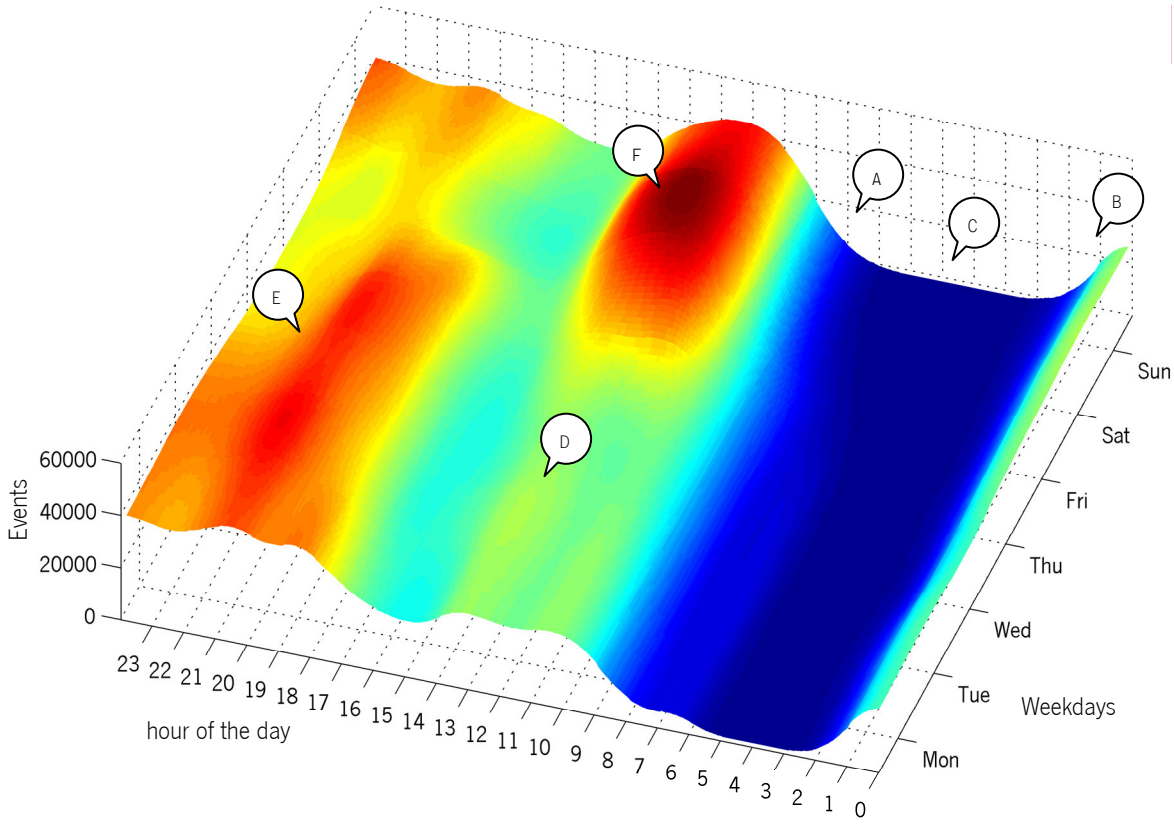


Figure 35 – Daily Accumulated Events per Weekday and per hour of the day.
Labels: A- Wake-up; B- Go to Sleep; C- Sleeping; D- Lunch; E- Dinner; F- Cleaning-up

The habits of the users (of the family studied in this project, in particular) tend to change over time, which inevitably changes the daily profile activities. One of the major contributions to

those changes are: the work place, working hours, the child school schedule or the type of professional projects in development.

Figure 36 presents the evolution of the daily profile over time. This figure shows the accumulated number of events during the day per weekday divided into four semesters, within the two-year period. In the figure, the dark red regions mean a greater value of accumulated number of events and the dark blue region a lower number of events.

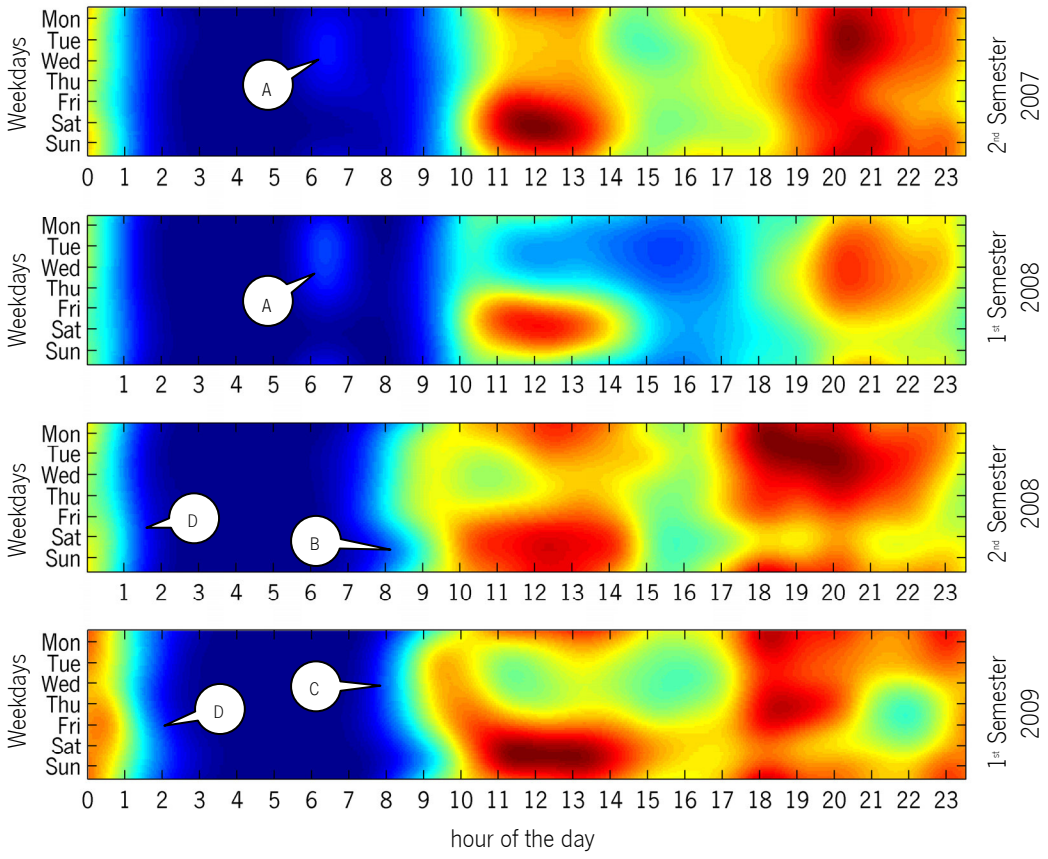


Figure 36 – Accumulated Events per Weekday and per hour of the day, divided by semester. Labels: A- Womans Wake-up; B- Weekend Wake-up Hour; C- Working Day Wake-up; D- Go to Sleep;

In Figure 36, it is noticeable that the second semester of 2007 is similar to the first semester of 2008 and the same happens between the second semester of 2008 and the first semester of 2009. These similarities can be explained by the fact that the woman of the family is a teacher. In the two upper semesters, a small activity is reported between 6 to 7 am which is the time the

woman gets up. Friday was her day-off, so that was when she worked at home, explaining the large activity on that day.

The getting-up time differences between the working day and the weekend days is very noticeable on the last two semesters (second semester of 2008 and first semester of 2009). The behaviour that occurs during the day also changes over time.

8.4 Inhabitants' daily routine through home light usage

The habits of the users in a home environment are related to their location inside the home, i.e., its inhabitants have conventionally different tasks in function of the room they are in. These activities, related to specific rooms in the house, unequivocally appears in the light usage of those particular rooms, especially during low day light levels and night (from 18pm to 1:30am).

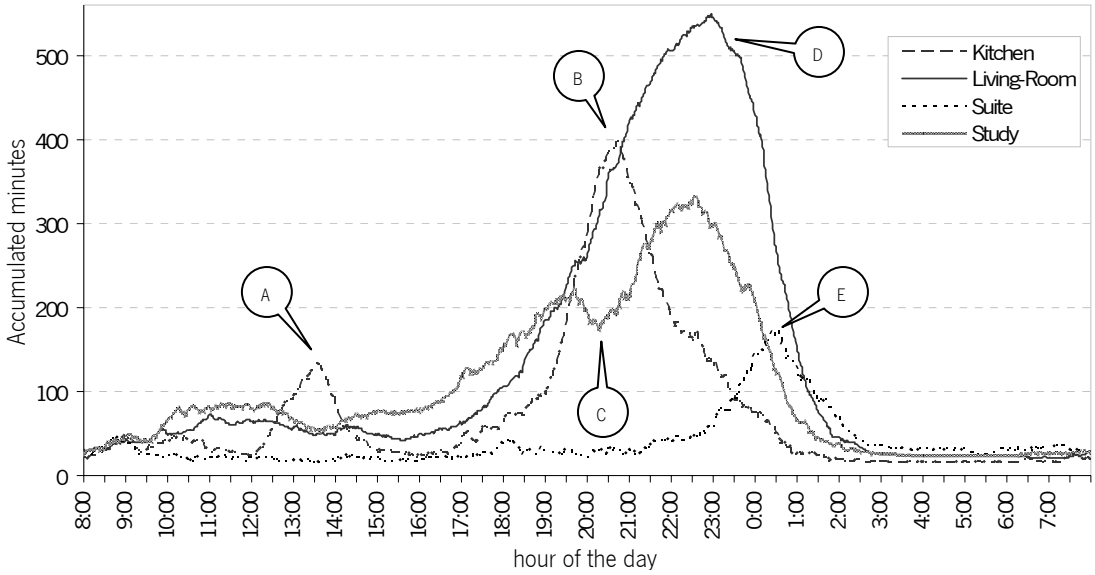


Figure 37 – Accumulated light usage per hour of the day during the two-year interval. Labels: A- Lunch; B- Dinner; C- Work Break; D- Living-Room (watching TV); E- Go to Sleep

In Figure 37 it is presented the accumulated light usage of four rooms of the flat (kitchen main light, living-room ambient light, suite and study). The values presented show the accumulated minutes during the day (with a minute resolution) where the lights were activated, throughout the two-year-study period.

In Figure 37 it is possible to observe some features, in the peak of the light accumulated usage, that represent the family daily routine. For example, in the light of the kitchen it is possible to distinguish that there are two noticeable peaks, one at 1:30 pm and another at 8:30 pm, presenting the lunch and dinner time, respectively. The difference in amplitude of the peaks in the kitchen light is explained by the light of the day and the night time.

Another interesting feature, presented in the light usage of the study room, is the depression that happens at the same time as the peaks in the light of the kitchen, indicating a break in the work that is being done in that particular room.

The living-room ambient light seems to be turned on at night, around 80% of the days in the two-year captured data and during long intervals, resulting in a relatively smooth curve. The family goes to sleep around 0:30 and it is when the suite light accumulated usage has its highest value.

The curves shown in Figure 37 can also give indication about some aspects of the light usage, for example: the total power consumption, the duration of activations and even the precision of events. The total power consumption can be calculated by integrating the accumulated values over the minutes of the day. The duration of the activation is noticed by rapid changes in the curves of the accumulated values. Some repetitive events with a very precise timing rapidly accumulate values that come up in the form of a very noticeable peak in the curves.

The histogram presented in Figure 38 shows that the activation in the living-room light (ambient light) has a period that lasts from 2h to 6h, comparing to the suite-room light whose activations have a duration from 360s to 1200s (6 to 20 minutes). This shows that in the living-room the light serves the users for activities that take longer time (e.g., watching TV) in contrast to the suite light activation that is used for actions that take less time (e.g. going to sleep).

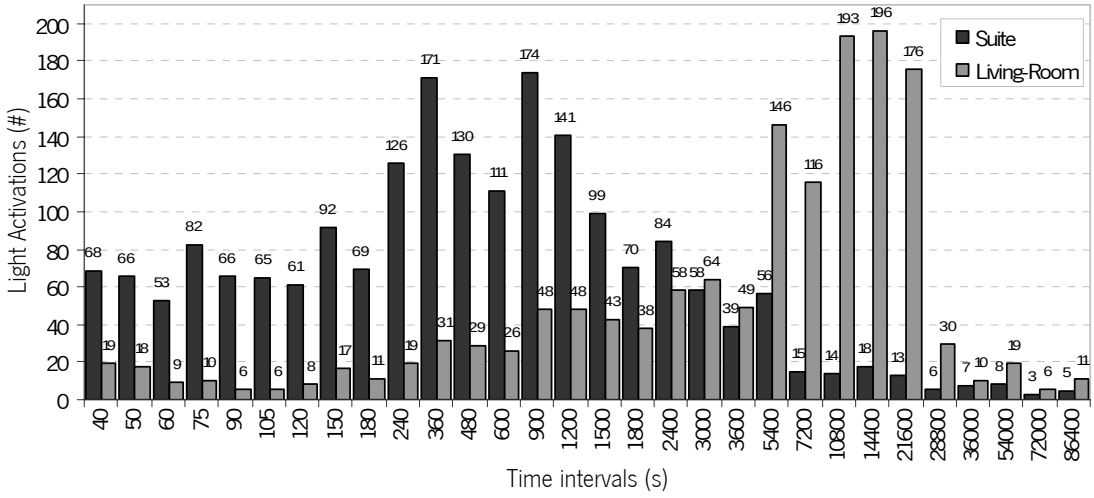


Figure 38 – Histogram of light usage period (suite and living-room) exponential scale

The WC main light usage is a good case-study to verify the routines without the interference of the day light. The flat WC does not get any direct sunlight and therefore any usage of that room requires light activation. The accumulated WC light usage, presented in Figure 39, shows that its use happens all over the day, accumulating relatively few minutes in the two-year study interval. It is also noticeable that the light in the WC was left on during the night in some days (12 days), as the values have an offset of 12 minutes during night hours.

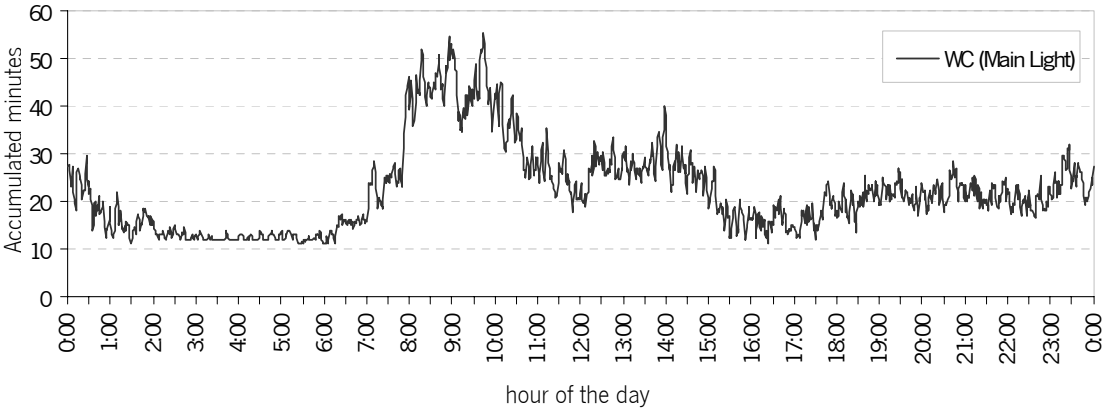


Figure 39 – Accumulated usage (in minutes) of WC main light across the day (two year interval)

The histogram of Figure 40 is presented in order to better understand the WC light usage. In the picture below it is possible to distinguish two inhabitants usage of the room, which is represented by the two bars, one at 60s (1 minute) and the other at 360s (6 minutes). The histogram percentile shows that 95% of the light usage periods have a value lower than 900s (15 minutes), meaning that the light activations with a duration above 15" are rare for the WC main light. A percentage of those rare and long activations have occurred when the users forgot to turn off the WC light.

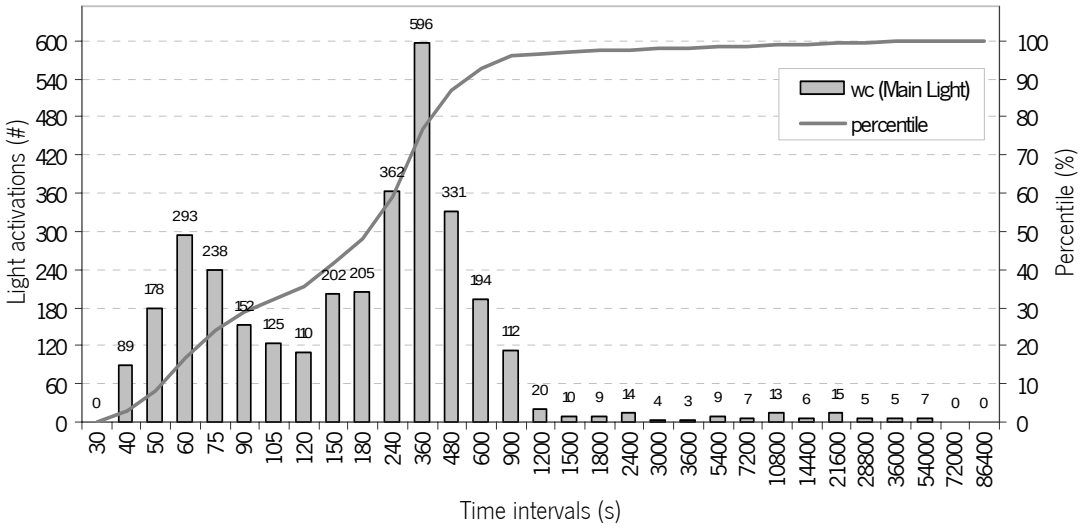


Figure 40 – Histogram of WC light usage period and histogram percentile

Within the two-year study interval, the WC light was activated a total of 3315 times, and therefore 5% (the last percentile left) represent 127 light activations. An autonomic action of turning off the light, only based on a single threshold of the histogram percentile, inevitably fails. The solution to this problem is not only to use the statistics, but also to complement them with artificial intelligence algorithms, as presented in the previous chapter or on the published paper [Machado et al., 2008].

The single usage of complex AI algorithms to create an autonomic decision by the system is not the ultimate solution. The AI algorithms need to be helped by simple statistics metrics and calculations. For example, in the WC light usage, it is possible to calculate the average activations

per day, which in this particular case has the value of 4,56 activations. The average of activations does not show if the activations are evenly distributed around the day or happen all together in the day.

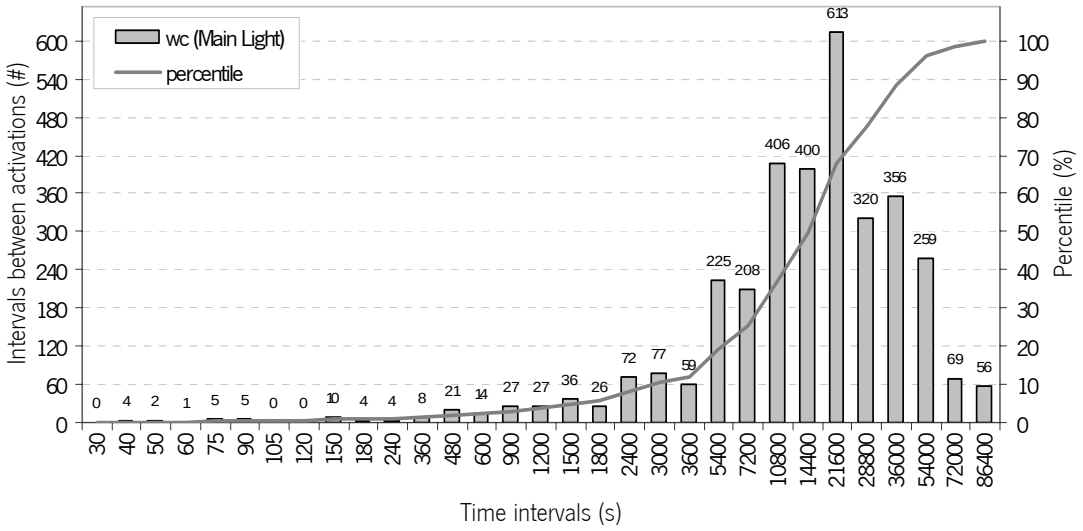


Figure 41 – Histogram of WC light intervals period between activations and histogram percentile

Figure 41 shows that the histogram of the WC light intervals period where activations tend to be separated from each another 14400s to 21600s (4h to 6h), meaning that the activations are spread all over the day evenly.

This histogram statistics can also support the AI algorithm in the decision to turn on the light automatically. When the AI algorithm decides to execute the action, it needs to be crosschecked according to intervals between light activations, enabling or disabling the final decision.

8.5 Light usage cost and potential power saving economy

The system analysed in the thesis focus on capturing the activities of the users activity in the light system of a flat. The habits of the users, reflected in the captured activities, can be used to create context-aware applications.

The light system of the flat can also benefit from the light usage patterns defined by the daily life of the inhabitants. An automatic deactivation of a light that was unwillingly left on reduced the energy used by the house which has an impact in the electricity bill of the flat.

As shown in Figure 42, the energy distribution in the histogram of light usage period has an exponential evolution. The energy usage can be reduced exponentially if the long activation periods of a light are avoided.

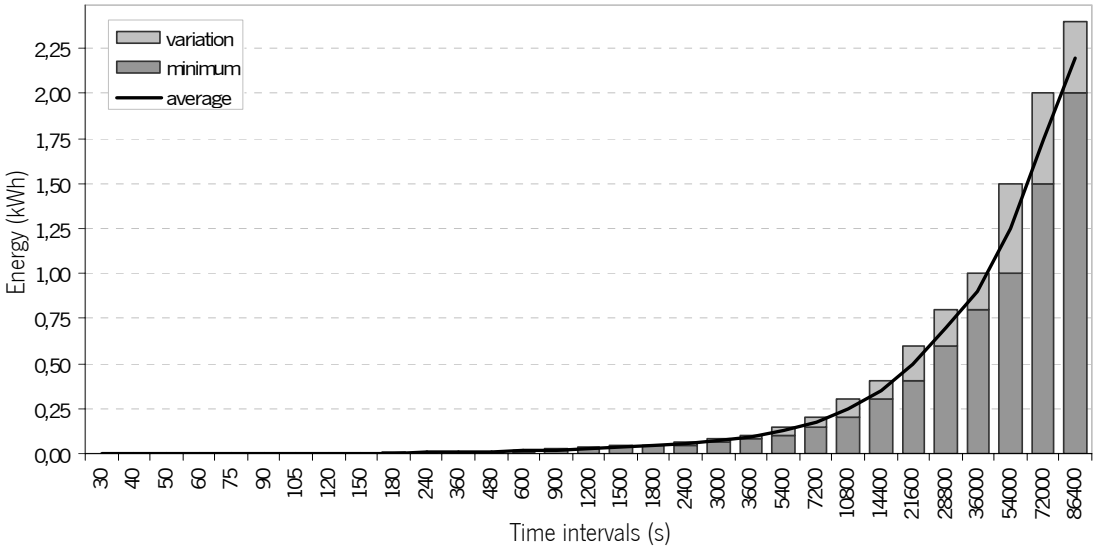


Figure 42 – Energy used (100W light) per histogram intervals

In order to understand the potential energy savings that may occur from an automatic action of turning-off the lights, it has been used the histogram percentile as the main indicator to the automatic decision.

Flat Division	Light Point	Power (W)	Total Light Usage			Light Usage (percentile>95%)			Light Usage (percentile>97,5%)		
			Time (h)	Energy (kWh)	Cost (€)	Time (h)	Energy (kWh)	Savings (€)	Time (h)	Energy (kWh)	Savings (€)
SUITE	Main	35	265,36	9,29	1,04 €	189,17	6,62	0,74 €	177,00	6,20	0,70 €
	Entrance	20	1088,54	21,77	2,45 €	700,00	14,00	1,57 €	544,00	10,88	1,22 €
WC SUITE	Main	10	445,29	4,45	0,50 €	328,67	3,29	0,37 €	288,33	2,88	0,32 €
	Mirror	50	962,19	48,11	5,41 €	557,42	27,87	3,13 €	427,17	21,36	2,40 €
BALCONY NORTH	Main	40	30,07	1,20	0,14 €	24,00	0,96	0,11 €	24,00	0,96	0,11 €
BEDROOM	Main	50	638,85	31,94	3,59 €	399,42	19,97	2,24 €	343,58	17,18	1,93 €
STUDY	Main	40	2909,97	116,40	13,08 €	1557,00	62,28	7,00 €	1023,00	40,92	4,60 €
CORRIDOR	Main	100	235,34	23,53	2,64 €	172,57	17,26	1,94 €	158,25	15,83	1,78 €
MAIN WC	Main	10	640,25	6,40	0,72 €	465,67	4,66	0,52 €	406,08	4,06	0,46 €
	Mirror	100	420,29	42,03	4,72 €	310,23	31,02	3,49 €	270,63	27,06	3,04 €
HALL	Main	50	459,94	23,00	2,58 €	344,50	17,23	1,94 €	301,50	15,08	1,69 €
ENTRANCE	Main	40	108,77	4,35	0,49 €	100,50	4,02	0,45 €	78,00	3,12	0,35 €
KITCHEN	Main	15	2099,73	31,50	3,54 €	1208,00	18,12	2,04 €	996,00	14,94	1,68 €
	Sink	40	2803,46	112,14	12,60 €	1554,00	62,16	6,98 €	1262,00	50,48	5,67 €
	Laundry	15	279,50	4,19	0,47 €	211,67	3,18	0,36 €	186,50	2,80	0,31 €
LIVING-ROOM	Main	50	539,08	26,95	3,03 €	267,00	13,35	1,50 €	203,00	10,15	1,14 €
	Dining Table	140	157,12	22,00	2,47 €	97,00	13,58	1,53 €	84,00	11,76	1,32 €
	Environment	35	3914,15	137,00	15,39 €	1879,00	65,77	7,39 €	577,00	20,20	2,27 €
	Indirect	100	394,11	39,41	4,43 €	246,00	24,60	2,76 €	171,00	17,10	1,92 €
BALCONY SOUTH	Main	40	1,05	0,04	0,005 €	0,50	0,02	0,002 €	0,50	0,02	0,002 €
			18393,06	705,70	**79,29 €**	10612,29	409,94	**46,06 €**	7521,54	292,96	**32,91 €**

Table 15 –Light usage savings (total percentile >95% and >97,5%) (1kWh = 0,11235 €)

In Table 15, it is presented the calculation of energy consumption by the lights, for the flat. The total cost was calculated, as well as the savings for light usage where the duration in the histogram has percentiles above 95% and 97,5%. It shows that the lights have been used for a total of 18393 hours, consuming more than 700 kWh of energy. The total cost was calculated, taking in account the power of each light bulb installed, costing almost 80€, considering the price of 1kWh of 0,11235€ (price/kWh that includes taxes).

Table 16 –Light usage cost potential gain with automatic turn-off actions

Light Usage	Time (h)	Energy (kWh)	Savings (€)	Total Cost (€)	Gain (%)
Total	18393,06	705,70	0,00 €	79,29 €	**0%**
Percentile > 95%	10612,29	409,94	46,06 €	33,23€	**58,0%**
Percentile > 97,5%	7521,54	292,96	32,91 €	46,38€	**41,5%**

The Table 16 shows that the potential savings for an automatic turn-off action of the lights are considerable. The values of potential savings were 58% and 41,5% for percentiles >95% and >97,5%, respectively. These savings are substantial because not all activations above were unintentional. However, the high values calculated show that the potential gains may be sufficient to risk a wrong action from the system.

Nevertheless, any wrong actions from the system must be avoided at all cost. They affect the confidence of the users in the system. In order to prevent a drastic action, such as turning-off all the lights in a room which completely leaves that space in the dark, the system may start a procedure that reduces the light over-time or changes the light source to a light bulb with less power consumption. This procedure can be used by the system to test the acceptance by the inhabitants of an automatic action.

8.6 Small Period Light Activations Phenomenon

During data analysis, it was observed that many light sources presented small period activations, i.e., the lights were activated and immediately (or in less than 2 seconds) deactivated.

Some activations are easily explained because of the way the system operates in order to change the light output power from ½ (half power light) to full power output or activating the light point from a different switch, using transition sequences on it. An example of this happens with the bedroom light whose system activates in half-power by default and only switches to full light-power if the user executes two consecutive transitions in the light switch. During the switch transitions, the bedroom light flickers from half-power to full-power, resulting in a high value of the light, as it can be noticed in Figure 43.

However, in the WC of the suite, highlighted in Figure 43 with bars with different filing, for each light point (main light and mirror light) that has an individual switch, the explanation to the phenomenon is different.

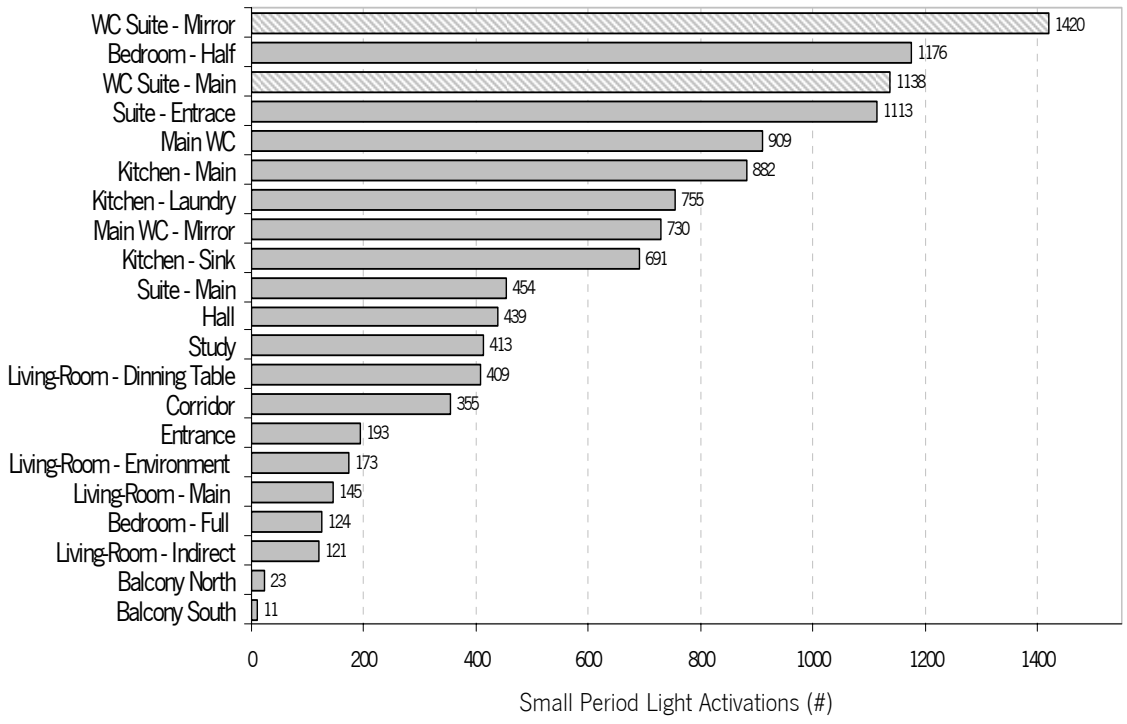


Figure 43 – Small Period Light Activations (with period ≤ 2 s) (two- year interval)

In this WC, the light operation was setup in a different way and in a non- standard way. Before the installation of the system, it was noticed that the WC lights tend to be used at the same time as in almost all WC usages. This situation was not acceptable in terms of power saving, because the mirror light tends to obfuscate the light produced by the low-power main light bulb. In order to avoid this situation and to get some energy saving, the system was setup not to allow the two light points to be activated at the same time. The setup, in the Teleswitch device, was defined to switch-off any lights in an exclusive way, i.e., when activating a light point, the other is turned-off at the same time.

The setup of the WC light in a non-standard way has immediately started to be noticeable by the users due to their strange behaviour. When activating the light, the difference was almost not perceived by the users, but whenever they tried to turn-off the lights, then the problems got started, mainly when they mistook the correct switch. When that happened, the other light turned on, the other turns-off and the user noticed that he has mistaken the switch. Immediately after, the user corrected the action by pressing the other switch which turned on the original light and turned off the other one. At this point the user is exactly in the same situation it had started. This sequence can happen consecutively unless the user concentrates on what is the right switch for the light that is supposed to be activated. This situation leads to some sort of frustration of the users concerning the setup of lights.

Nevertheless, as time went by, the users have changed their behaviour to avoid the cycle of mistaken actions. One of the users reported that he always uses the same light (mirror light), using therefore the same switch. The other user uses employs the two lights, but when turning off the light, it executes a rapid switch-on and switch-off in the same switch that results in all lights being turned off.

The difficulty of the user to turn-off the light in the WC shows that simple changes in the traditional functionality may have impact on a non-foreseen way.

8.7 Tools Used for Data Processing

During this project, a software application was developed in order to manage a large amount of captured data. This application, called USTAT (UDOMUS Statistics) shown in Figure 44, organizes the data captured in a DB, performs some calculations and allows a graphical representation of the processed data. It was also used as a testing application of the calculation needed to an autonomic action, helping in the analysis and in the understanding of system actions.

The processing operations implemented in the software include: the calculation of room light usage periods, an histogram of the light usage, an histogram of time among usages and a daily light profile usage for every minute. Other statistics that support AI algorithms are also included in this software.

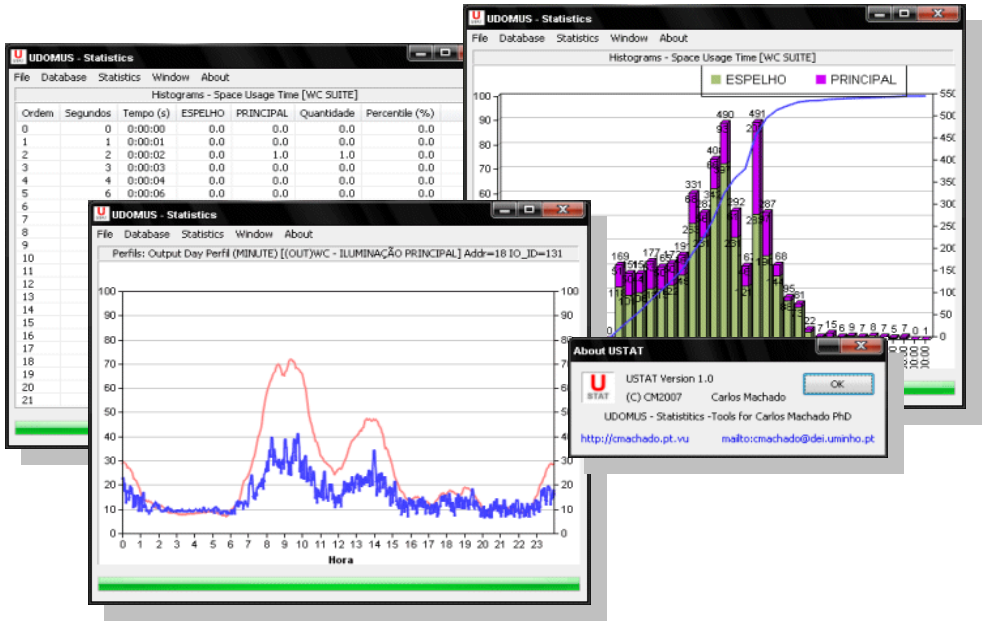


Figure 44 – USTAT Application – screenshots

Other conventional and standard tools have been used to process data and create the graphics, e.g., MATLAB, Excel and MS Access. The access SQL queries were a very important tool

because it allowed the manipulation of the huge amount of events that were captured during this experiment.

8.8 Summary and Conclusions

The analysed data show that the performance of the system for event capturing was appropriate to the ubiquitous systems requirements, i.e., it was possible to capture data that is related with the daily activities of a family without affecting their daily routines for a long period of time.

During the two-complete year time interval experiment, it was possible to identify some important dates to the family living in the flat. Those dates were detected because the family performed activities that are not normally done on a typical day. These special days produce an accumulated value of events that do not match the standard values captured from the family typical day, i.e., they produced much more events or much less than on an average day.

In general, the inhabitants´ daily captured activity flows in a normal distribution or in a Gaussian distribution, which can be used to determine an usability index that may be used in higher level context-aware applications. The usability index can even become more precise if it uses the information of the current period of the year and even the information of every weekday.

The data captured also shows that users tend to repeat activities annually and in specific periods of the year. This fact enforces the idea that humans are in fact creatures of habits, even when considering long periods (e.g a year).

The daily routines throughout the weekday also show patterns, and it is very noticeable the difference between working days from the weekends. It was also possible to verify that the workday daily routine tend to change over time according to inhabitants job requirements.

The capture of the flat rooms light usage allows the determination of the sequence of activities executed by inhabitants throughout the day. That captured information can be used to create an application based on context, but at the same time it can be used to improve the energy efficiency of the flat. The system can determine the light daily usage profile and it is possible to determine the moment to turn-off a light when it has been forgotten by a user. The ability to shut-down lights automatically, those that are activated outside their normal usage, can reduce the energy consumption in values round 50%.

According to the analysis of the captured data, it is also shown that is possible to determine some inadaptability of the users in a particular setup. It is also possible to determine some potential component failure, verifying strange events or by abrupt change in the inhabitants´ behaviour.

The capture of light usage and the movements around the flat show that it is possible to create a context able to the potential assist the inhabitants in a non-obstructive way. The sensibility of having captured personal data is not problematical if it is used locally to improve the quality of life of the users. Further, the potential benefits of capturing simple actions daily are remarkable.

9 Conclusions

In this thesis it is described an approach to the construction of an autonomic ubiquitous system. This type of system, which brings together many fields of computing, has the purpose of creating a computing system that is reflexive and self-adaptive. AI techniques can be used to achieve a system with such adaptability. Although the modern AI techniques have shown to be a solution to build adaptable automation, it was raised in the beginning of this thesis that this type of systems, which are deployed in the environment, require other capabilities in order to be successfully applied.

The technology to be used in a ubiquitous system requires some characteristics in order to meet the challenges enumerated in the introduction of this thesis. The challenges require a global perspective view of the systems as well as a local perspective view. A system global diagram, representing the way the main components may interact was defined, and all the components that must exist in this type of systems were described. The main system design characteristics, i.e.: distributed, fault-tolerant, impromptu interoperable and scalable, were listed as the guidelines to be taken into account when planning the implementation of robust system components.

In order to increase the robustness of the system, the design characteristics proposed in this thesis use the concept of reflexive actions and actuators feedback events (action feedback events). The reflexive actions allow the actuators to perform predefined actions in an autonomous mode and without the intervention of higher level system components. The action feedback events feature allow the higher level system components to perform the necessary system monitoring, i.e., allowing the measurement of the correct operation of the systems by monitoring and predicting their behaviour.

After the definition of the philosophy that guides the architecture of the system, the attention was focused on the details of the low-level components (sensors, actuators and communicators). The guidelines, characteristics and concepts that have been used to the implementation of these components were published in the paper [Machado et al., 2007].

The human ANS reflex arc was used as the inspiration of the capabilities required by the low level components of an autonomic ubiquitous system. The reflex arc can execute an action without the intervention of the human brain, resulting in a fast response time. The analogy also specifies that the system low-level components need to be able to execute any basic function of the system. This was achieved by providing the component with three main capabilities: event-driven information, the transmission information in broadcast mode and configurable binding among events and actions. These characteristics, linked to the ability to monitor the internal variables, of devices allow them to be used in the upper system layers.

In order to test the concept proposed in this thesis, the Teleswitch device was built. This development helped to clarify some technical aspects that need to be overcome to make its implementation and possible deployment in a human environment. This device was also implemented using hardware traditional communication technology, which simplifies the design, resulting therefore in a low cost device. The usage of traditional technology means that it is not necessary to adopt a state of the art communication interface in devices, showing that the philosophy is the most important aspect in this ubiquitous system design.

In a ubiquitous system, the humans are a part of the system loop. In order to validate the theory proposed in this thesis, a home environment was selected. A system was installed and allowed the confirmation that the required capabilities presented earlier in this thesis were suitable to the implementation of devices for this type of systems. The quality of the data that has been acquired for a large period of time shows that the setup was appropriate. The captured data shows the ability to infer the behaviour of the inhabitants, allowing the extraction of patterns of the inhabitants during a year, as well as, during a day. The patterns presented show potential benefits to the user as well to the optimization of the system functionality. In particular, it has been shown that the energy economy in a home light system can be reduced 50%. The results reveal a relevant aspect in today's society concerning energy savings.

As sooner as the system installation and the capturing of events started, the focus of the work of the thesis was the study of Artificial Intelligence. The AI is necessary to a system to be adaptable to the patterns of the users, i.e., a context-awareness construction in order to create useful applications that react to it. The Artificial Neural Networks (ANN) approach was selected,

although many other methods and algorithms may have been used in order to bring intelligence to autonomic actions. The ANN ability to detect patterns in a large and noisy dataset, reducing it to a relatively small amount of data (ANN weights) and in combination with unsupervised learning, makes it a practical and ideal solution. This thesis presented a case-study where the ANN was trained with real data to determine the state of the lights in the home. The results have showed that although the ANN has learnt the light usage pattern, it still fails in some situations. It has also indicated that any autonomic action can not only be decided with the ANN and that basic statistics must be used to introduce some certainty to the autonomic action. The method used and its results were published in the paper [Machado et al. 2008].

The system user feedback and control is another important aspect that must be always present in this type of systems. The user interface allows the user to understand the system, by creating a mental model about how it works. For that reason, the system must always have some type of visualization and manual overwrite control in order to allow the users to intervene in the system, if necessary.

9.1 Contributions

The work presented in this thesis tackles some particular aspects in the construction of autonomic ubiquitous systems, defining several system design requirements that need to be used for practical implementations.

The definition of a global system structure, which can be used as a template in the design of this type of system, is an important part because it provides an overview of the main components and layers of the system. The identification of each block with generic inputs and dataflow helps in the definition of those essential requirements.

Another important characteristic in the design philosophy presented in this thesis is the concept that devices in the environment should broadcast information, i.e., the sensors and actuators embedded in the environment, broadcast to all potential listening devices event

information that reflects the users´ actions or internal state changes. This characteristic allows the systems to be included in a newer installation, because the essential environment monitoring is preserved.

The design definition that devices and systems deployed in the environment should perform reflexive actions, in order to carry out the basic functionalities, was presented in this thesis. The reflexive actions were achieved using simple configuration definitions (namely configurable binding tables) in combination with the events that are available in the data bus. These reflexive actions results in a faster response by the systems, increasing at the same time its robustness, because basic or essential functions do not require the upper-layer-devices intervention. The upper layer components can use the device simple configuration definition capability to embed behaviour, making it a part of the environment. This can be done using the behaviour patterns determined by analyses of the captured event data that are available on the data bus.

In this thesis, it has also been presented the essential capabilities, together with the obvious ability to communicate, that must exist in hardware devices and system components. These capabilities allow the implementation of data broadcasting, binding of events into actions and the execution of reflexive actions, making them a part of an autonomic ubiquitous system.

Concerning the artificial intelligence of the system and usage of the context-awareness, a method has been devised for the determination of the state of the lights in a house that uses the ANN as a classifier algorithm in combination with basic statistics. This method uses the actions executed by users, in an implicit way, and without the need to perform data labelling.

Based in a real installation, the captured data allow the calculation of the gains that an autonomic action could have in saving energy. The analyses show that the ability to shut-down lights automatically, those that are activated outside their normal usage, can obtain a reduction of energy consumption around 50%.

9.2 Future Work

Obviously, the work presented in this thesis is far from being completed, however, based on the achievements attained so far: Teleswitch devices, communicator device, testing environment and a collection of data throughout a long period of time, further studies can be carried out using these resources. Using the currently available setup it is possible now to focus further research on the higher level functionalities of the system and enhancing the installation with more features in order to improve the overall experience of such ubiquitous system.

The testing of different AI algorithms or ANN with a more complex structure is a type of study that can be carried out in the already available dataset. The results of classification and prediction of the behaviour of the users can be compared, pointing out the advantages and disadvantages of each solution.

When the system starts to actively intervene in the environment, by executing an automatic action, there are potential changes in the behaviour of the users. These modifications can have severe negative impact on the acceptance by users of the system. The study of these transformations in the habits of the users, promoted by the system, is also a very interesting subject of research in order to define the best guidelines for user interface design.

The philosophy proposed in this thesis requires that data is sent in broadcast mode every time an event occurs. This method of operation can have a negative impact on energy consumption, particularly in battery powered devices, because of this limited resource. The studies on negative impacts versus potential benefits need to be thoroughly studied. Some flexibility in the philosophy may be necessary in this class of devices, so a balance between system functionality and energy consumption can be achieved.

The architecture that has been designed and developed throughout this project proved to be very robust and ready to be installed in any other environments, in particular in other homes. However, installation in new homes with a different family structure might reveal other interesting features and enforce new conclusions.

In the installation environment reported in this work, only the house light system was instrumented. This limits the ability to detect more user patterns and also imposes some restrictions in the ability to detect the energy consumption of the studied home. The capturing of more users` actions at their home is the next step. These actions are particular activities to be performed in the kitchen (usage of the fridge, cooker, washing machine and dishwasher), in the living-room (by capturing the devices remote control coded IR emissions) and bedrooms (alarm-clock usage). The window shutters usage and control is another house system that may complement the inhabitants' activities around the house. The measurement of water, energy and gas consumption should also be considered as another source of relevant information for extraction of patterns of usage by users. To achieve the complete energy usage profile, the house climate system also needs to be instrumented (heating system or air conditioning). With the complete instrumentation of the house, evident benefits in the energy efficiency usage helps this type of technology to be fully deployed in the market.

The feedback and control of the system by users is also supposed to be available throughout the home. At this moment, this is already possible, by using the PDA with WI-FI connection, but users tend not to use it when they are at home. In the home environment, the feedback and control using the already available TVs and its remote controls is the solution to spread out the feedback of the system. The TV may display running messages or a small symbol over normal TV emission images. This requires a device that generates a video overlay image and captures the IR coded data from the TV remote control.

9.3 Concluding Remarks

The privacy of people is an ever growing concern that is associated with the existence of systems that capture more and more aspects of personal daily activities. This aspect is more significant when it happens in a public place and the captured information nature is unknown, and at the same time, not controlled by users. The feeling of being watched and the actions reported to a "big-brother" like system can be unacceptable to some users. In home environments, where the users own and control the captured data, the concern changes from privacy to security. Users

should be able to choose how that data is used, the period of time that is kept. The outside control and access to the data should be avoided, by making only available to the outside the indicators that are authorised by the users.

Users of this type of systems need to be aware that the system will learn there behaviour in order to reproduce it later. In this way the system will reproduce the good user behaviours as well as bad user behaviours. If users want the system to perform in a particular way, they need to give to the system, day after day, good patterns, e.g., if a user desires that the system switches off the lights that were left on, he needs to avoid leaving the lights on. In this way, if a user exceptionally leaves a light activated, the system will not have any problems detecting it and consequently switch off the correspondent light.

The autonomic ubiquitous system uses information gathered without a direct intervention of the user and uses this information to act in the environment, estimating what the user is thinking in that particular circumstance. This idea is very difficult to be achieved by a machine and even by any human, without some degree of error. In conclusion, any decision made by the system to not act in a particular situation is always preferable to an erroneous automatic action.

References

1. [Weiser, 1991] M. Weiser., The Computer for the 21st Century. Scientific American, 265(3): 94-104, 1991.

2. [Weiser and Brown, 1995] M. Weiser and J. S. Brown. Designing Calm Technology. December 1995.

3. [Ishii and Ullmer, 1997] H. Ishii and B. Ullmer. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In Conference on Human Factors in Computing Systems (CHI), 1997.

4. [Schilit et al., 1994] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In Workshop on Mobile Computing Systems and Applications, 1994.

5. [Brown, 1996] P. J. Brown. The Stick-e Document: a Framework for Creating Context-aware Applications. Electronic Publishing, pages 259–272, 1996.

6. [Abowd et al., 1997] G. D. Abowd, A. Dey, R. Orr, and J. Brotherton. Contextawareness in Wearable and Ubiquitous Computing. In International Symposium on Wearable Computers (ISWC), 1997.

7. [Abowd et al., 1999] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a Better Understanding of Context and Context- Awareness. In International Symposium on Handheld and Ubiquitous Computing (HUC), 1999.

8. [Antifakos et al., 2003] S. Antifakos, J. Borchers, and B. Schiele. Designing Physical Interaction with Sensor Drawbacks in Mind. In Physical Interaction, Workshop at MobileHCI, 2003.

9. [Schmidt, 1999] A. Schmidt. Implicit Human-Computer Interaction through Context. In International Symposium on Human Computer Interaction with Mobile Devices and Services (MobileHCI), 1999.

10. [Edwards et al., 2001] W. Edwards, R. Grinter, “At Home with Ubiquitous Computing: Seven Challenges,” Proceedings of the Ubiquitous Computing 2001 (UBICOMP 2001), Springer Verlag, LNCS 2201, 2001, pp. 256–272.

11. [Abowd et al., 2000] Abowd, G.D.,Mynatt, E.D.: “Charting Past, Present, and Future Research in Ubiquitous Computing.” ACM Transactions on Computer-Human Interaction, 7 (1). (2000) 29-58

12. [Kephart et al., 2003] Kephart J. O., Chess D.M.. "The Vision of Autonomic Computing". Computer, IEEE, Volume 36, Issue 1, January 2003, Pages 41-50

13. [McCann, 2003] McCann J .A. "The Database Machine: Old Story, New Slant?" Proceedings of the first Biennial Conference on Innovative Data Systems Research, VLDB, January 5-8 2003

14. [Pescovitz, 2002] Pescovitz D. "Helping computers help themselves." Spectrum, IEEE, Volume 39, Issue 9, September 2002, Pages 49-53

15. [Markl et al., 2003] Markl V., Lohman G. M., Raman V., “LEO: An autonomic query optimizer for DB2.” IBM Systems Journal, Vol. 42, No. 1, 2003.

16. [Kidd et al., 1999] C. D. Kidd, R. J. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner and W. Newstetter. "The AwareHome: A Living Laboratory for Ubiquitous Computing Research", Proc. of the Second International Workshop on Cooperative Buildings, 1999.

17. [Robert et al., 2000] Robert J. Orr and Gregory D. Abowd.,"The smart Floor: A mechanism for natural user identication and tracking." In Proceedings of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, Netherlands, April 2000. ACM.

18. [Patel et al., 2006] S.N. Patel, K.N. Truong, and G.D. Abowd, “Powerline Positioning: A Practical Sub-Room-Level Indoor Location System for Domestic Use”, Proc. Eighth Ann. Conf. Ubiquitous Computing (UbiComp '06), pp. 441-458, 2006.

19. [Shehan et al., 2007] Erika Shehan and W. Keith Edwards., “Home Networking and HCI: What Hath God Wrought?”, Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'07). San Jose, CA. April 28-May 3, 2007.

20. [Yang et al., 2007] Jeonghwa Yang and W. Keith Edwards.,“ICEbox: Toward Easy-to-Use Home Networking.”,Proceedings of Eleventh IFIP Conference on Human-Computer Interaction (Interact). Rio de Janeiro, Brazil. September 10-14, 2007.

21. [Cook et al., 2003] D. J. Cook, M. Youngblood, E. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: An Agent-Based Smart Home", Proceedings of the IEEE International Conference on Pervasive Computing and Communications, pages 521-524, 2003.

22. [Roy et al., 2003] A. Roy, S. Bhaumik, A. Bhattacharya, K. Basu, D. J. Cook, and S. Das, “Location Aware Resource Management in Smart Homes”, Proceedings of the Conference on Pervasive Computing, 2003.

23. [Intille et al., 2004] S. S. Intille, K. Larson, J. S. Beaudin, J. Nawyn, E. Munguia Tapia, P. Kaushik, "A living laboratory for the design and evaluation of ubiquitous computing technologies," in Extended Abstracts of the 2005 Conference on Human Factors in Computing Systems . New York, NY: ACM Press, 2004.

24. [Intille et al., 2003] S.S. Intille, E. Munguia Tapia, J. Rondoni, J. Beaudin, C. Kukla, S. Agarwal, L. Bao, and K. Larson., ”Tools for studying behavior and technology in natural settings.” Proceedings of UBICOMP 2003, volume LNCS 2864. Springer, 2003.

25. [Tapia et al, 2004a] E. M. Tapia, N. Marmasse, S. S. Intille, and K. Larson,"MITes: Wireless portable sensors for studying behaviour", Proceedings of Extended Abstracts Ubicomp 2004: Ubiquitous Computing, 2004.

26. [Intille et al., 2006] S. S. Intille, K. Larson, E. Munguia Tapia, J. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson, "Using a live-in laboratory for ubiquitous computing research", Proceedings of PERVASIVE 2006 , vol. LNCS 3968, K. P. Fishkin, B. Schiele, P. Nixon, and A. Quigley, Eds. Berlin Heidelberg: Springer-Verlag, 2006, pp. 349-365.

27. [Garlan et al., 2002] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Towards distraction-free pervasive computing", IEEE Pervasive Computing 21 (2002), no. 2, 22– 31.

28. [Satyanarayanan, 1996] M. Satyanarayanan, "Mobile Information Access", "Mobile Information Access", IEEE Personal Communications, February 1996

29. [Judd et al., 2003] G. Judd, P. Steenkiste,"Providing Contextual Information to Pervasive Computing Applications" IEEE International Conference on Pervasive Computing (PERCOM), Dallas, March 23-25, 2003.

30. [Roman et al., 2002] Manuel Roman, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell and Klara Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces", In IEEE Pervasive Computing, pp. 74-83, Oct-Dec 2002.

31. [Chetan et. al., 2005] Shiva Chetan, Anand Ranganathan, Roy Campbell. “Towards Fault Tolerant Pervasive Computing”, In IEEE Technology and Society, Volume: 24, No. 1, pp 38-44, Spring 2005

32. [Greenhalgh et. al., 2001] Chris Greenhalgh, Shahram Izadi, Tom Rodden, Steve Benford, "The EQUIP Platform: Bringing Together Physical and Virtual Worlds", Technical Report, University of Nottingham, 2001. Available on-line at http://www.crg.cs.nott.ac.uk/~cmg/Equator/Downloads/docs/equip-final.pdf [2008]

33. [Villar et al, 2007] Nicolas Villar, Hans Gellersen, "A malleable control structure for softwired user interfaces", in Tangible and Embedded Interaction conference proceedings (TEI '07), pp. 49-56, 2007, ISBN:978-1-59593-619-6

34. [Gaver et al., 2004] William W. Gaver, John Bowers, Andrew Boucher, Hans Gellersen, Sarah Pennington, Albrecht Schmidt, Anthony Steed, Nicolas Villar, Brendan Walker, "The drift table: designing for ludic engagement", Extended abstracts of the 2004 conference on human factors and computing systems (CHI '04). Vienna, Austria. ACM Press. ISBN:1-58113-703-6.

35. [Chen et al., 1999] D. Chen, A. Schmidt, H.-W. Gellesen, "An Architecture for Multi-Sensor Fusion in Mobile Environments", In Proceedings International Conference on Information Fusion, Sunnyvale, CA, USA, July 1999.

36. [Gellersen et al, 2000] Hans-W. Gellersen, Michael Beigl and Albrecht Schmidt, "Sensor-based Context-Awareness for Situated Computing", Workshop on Software Engineering and Pervasive Computing SEWPC00 at ICSE 2000, Limerick, Ireland, June 2000.

37. [Beigl et al., 2001] M. Beigl, H.-W. Gellersen, and A. Schmidt, "MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects", Computer Networks, vol. 35, no. 4, Mar. 2001, pp. 401-409.

38. [Gellersen et al., 2002] H.W. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in mobile devices and smart artefacts", Mobile Networks and Applications 7 (2002), 341–351.

39. [Addlesee et al., 2001] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, Andy Hopper. Implementing a Sentient Computing System. IEEE Computer Magazine, Vol. 34, No. 8, August 2001, pp. 50-56.

40. [Harter et al., 2002] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward and Paul Webster "The anatomy of a Context-Aware Application", In Wireless Networks, Vol. 8, pp. 187-197, February 2002.

41. [Åkesson et al., 2002] Åkesson, K-P., Bullock, A., Greenhalgh, C., Koleva, B. and Rodden, T. "A Toolkit for User Re-Configuration of Ubiquitous Domestic Environments" in Companion to Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology, Paris, ACM Press

42. [Crabtree et al., 2002] Crabtree, A., Hemmings, T. and Rodden, T. (2002) "Pattern-based support for interactive design in domestic settings", Proceedings of the 2002 Symposium on Designing Interactive Systems, pp. 265-276, London: ACM Press.

43. [Rodden et al., 2004a] Rodden, T. Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Åkesson, K-P and Hansson, P., "Between the dazzle of a new building and its eventual corpse: assembling the ubiquitous home", Proceedings of the 2004 ACM Symposium on Designing Interactive Systems, August 1st-4th, Cambridge, Massachusetts: ACM Press.

44. [Rodden et al., 2004b] Rodden, T., Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Åkesson, K-P. and Hansson, P. (2004) "Configuring the ubiquitous home", Proceedings of the 6th International Conference on Designing Cooperative Systems, May 11th-14th, French Riviera: IOS Press.

45. [Lalis et al., 2003] Lalis, S., Savidis, A., & Stephanidis, C. (2003, January). "Wearable Systems for Everyday Use", ERCIM News, Special Theme: Embedded Systems, 52, 16-17.

46. [Want et al., 1992] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system", ACM Transactions on Information Systems (TOIS) 10 (1992), 91–102.

47. [Lamming et al., 1994] M. Lamming and M. Flynn, "Forget-me-not: Intimate computing in support of human memory", Proceedings of FRIEND21'94, the International Symposium on Next Generation Human Interface, February 1994.

48. [Mozer et al., 1995] Mozer, M. C., Dodier, R. H., Anderson, M., Vidmar, L., Cruickshank III, R. F., & Miller, D. (1995). “The neural network house: An overview.”, In L. Niklasson & M. Boden (Eds.), Current trends in connectionism (pp. 371-380). Hillsdale, NJ: Erlbaum.

49. [Mozer, 1998] Mozer, M. C. (1998). The neural network house: “An environment that adapts to its inhabitants.”, In M. Coen (Ed.), Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments (pp. 110-114). Menlo, Park, CA: AAAI Press.

50. [Mozer, 1999] Mozer, M. C. (1999). ''An intelligent environment must be adaptive'', IEEE Intelligent Systems and their Applications, 14(2) , 11-13.

51. [Mozer, 2005] Mozer, M. C. (2005). “Lessons from an adaptive house”, In D. Cook & R. Das (Eds.), Smart environments: Technologies, protocols, and applications (pp. 273-294). Hoboken, NJ: J. Wiley & Sons.

52. [Weiser, 1993a] Mark Weiser, ''Some Computer Science Problems in Ubiquitous Computing,'' Communications of the ACM, July 1993.

53. [Weiser, 1993b] M. Weiser. “Hot Topics: Ubiquitous Computing.” In IEEE Computer, October. 1993.

54. [McCann et al., 2004] McCann J .A., Huebscher M.C., ”Evaluation Issues in Autonomic Computing “, in the International Workshop on Agents and Autonomic Computing and Grid Enabled Virtual Organizations (AAC-GEVO’04), 3rd International Conference on Grid and Cooperative Computing Wuhan, China, 21-24 October 2004.Springer-Verlag Heidelberg, p. 597

55. [Antifakos, 2005] S. Antifakos. “Improving Interaction with Context-Aware Systems.”, PhD thesis, Dept. Computer Science, ETH Zürich, Switzerland, 2005.

56. [Purves, 2004] Purves, Neuroscience: Third Edition. Massachusetts, Sinauer Associates, Inc.,2004 , ISBN 0-87893-725-0

57. [Huebscher et al., 2005] Huebscher M.C., McCann J.A. “An Adaptive Middleware For Context-Aware Applications”, published by Personal and Ubiquitous Computing Journal, Springer 2005

58. [Lee et al., 2003] C. Lee, D. Nordstedt, and S. Helal. Enabling smart spaces with OSGi. Pervasive Computing, 2(3):89-94, 2003, ISSN: 1536-1268

59. [Weiser and Brown, 1996] M. Weiser and J. S. Brown., “The Coming Age of Calm Technology”, Xerox PARC, October 5, 1996

60. [Wisnesk et al., 1998] Wisneski, G., Ishii, H., Dahley, A., Gorbet, M., Brave, S., Ullmer, B. and Yarin, P., "Ambient Display: Turning Architectural Space into an Interface between People and Digital Information.", In Proceedings of the First International Workshop on Cooperative Buildings (CoBuild’98), Darmstadt, Germany, Springer-Verlag Heidelberg, p. 22-32, February 1998.

61. [Gellersen et al., 1999] Gellersen, H.W., Beigl, M., “Ambient Telepresence: Colleague Awareness in Smart Environments”, 1. Intl. Workshop on Managing Interactions in Smart Environments (MANSE 99), Dublin, Irland, Dec 1999 & Springer Verlag: Managing Interactions in Smart Environments, P.Nixon, G.Lacey, S.Dobson ed, pp 80-88, 1999.

62. [Machado et al., 2007] Carlos Machado, José Mendes, “Sensors, Actuators and Communicators When Building a Ubiquitous System” Paper presented on the conference IEEE ISIE 2007, in Vigo, Spain, 2007, ISBN:1-4244-0755-9 (p1530-1535).

63. [Lima et al., 2000] Pedro Lima, Andrea Bonarini, Carlos Machado, F. Marchese, C. Marques, António Fernando Ribeiro, “Omni-directional catadioptric vision for soccer robots”, Appeared in special issue on the EuroRoboCup of the Jornal of Robotics and Autonomous Systems, Elsevier, Amesterdam, 30 December 2000.

64. [Svaizer et al., 1997] Svaizer, P., Matassoni, M., Omologo, M., "Acoustic Source Location in a Three-dimensional Space using Cross-power Spectrum Phase", Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP97), Munich, Germany, April 1997.

65. [Cook et al. 2003] D. J. Cook, M. Youngblood, III E. 0. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: An agent-based smarthome", in Proc. of First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), 2003, pp. 521-524.

66. [Kautz et al. 2003] H. Kautz, 0. Etzioni, D. Fox, and D. Weld, "Foundations of assisted cognition systems", University of Washington, Computer Science Department, Technical Report, Tech. Rep., 2003.

67. [Tapia et al. 2004b] E. Munguia Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home setting using simple and ubiquitous sensors", in Proceedings of PERVASIVE 2004, vol. LNCS 3001, A. Ferscha and F. Mattern, Eds. Berlin Heidelberg: Springer-Verlag, 2004, pp. 158-175

68. [Rivera-Illingworth et al. 2006] F. Rivera-Illingworth, V. Callaghan, and H. Hagras, "Automated Discovery of Human Activities inside Pervasive Living Spaces", Proc. of 1st International Symposium on Pervasive Computing and Applications, 2006, pp. 77- 82.

69. [Lühr et al. 2007] S. Lühr, G. West, and S. Venkatesh, "Recognition of emergent human behaviour in a smart home: A data mining approach", Pervasive Mob. Comput. 3, 2, pp 95-116, 2007.

70. [Zheng et al. 2008] Huiru Zheng, Haiying Wang,Norman Black, "Human Activity Detection in Smart Home Environment with Self-Adaptive Neural Networks", in IEEE International Conference Networking, Sensing and Control, 2008., (ICNSC 2008), ISBN: 978-1-4244-1685-1, pp 1505-1510

71. [Machado et al. 2008] Carlos Machado, José Mendes, “Automatic Light Control in Domotics using Artificial Neural Networks”, Paper presented on International Conference on Embedded and Ubiquitous Computing (ICEUC 2008), in Venice, Italy, 2008, ISSN: 2070-3740 pp 813-818

72. [Polikar, 2006] Polikar R., ”Pattern Recognition”, In Wiley Encyclopedia of Biomedical Engineering”, Ed. Akay, M., New York, NY: Wiley., 2006.

73. [Beck et al., 1986] Beck, J. R., and Schultz, E. K. 1986. The use of ROC curves in test performance evaluation. Arch Path01 Lab Med 110:13-20.

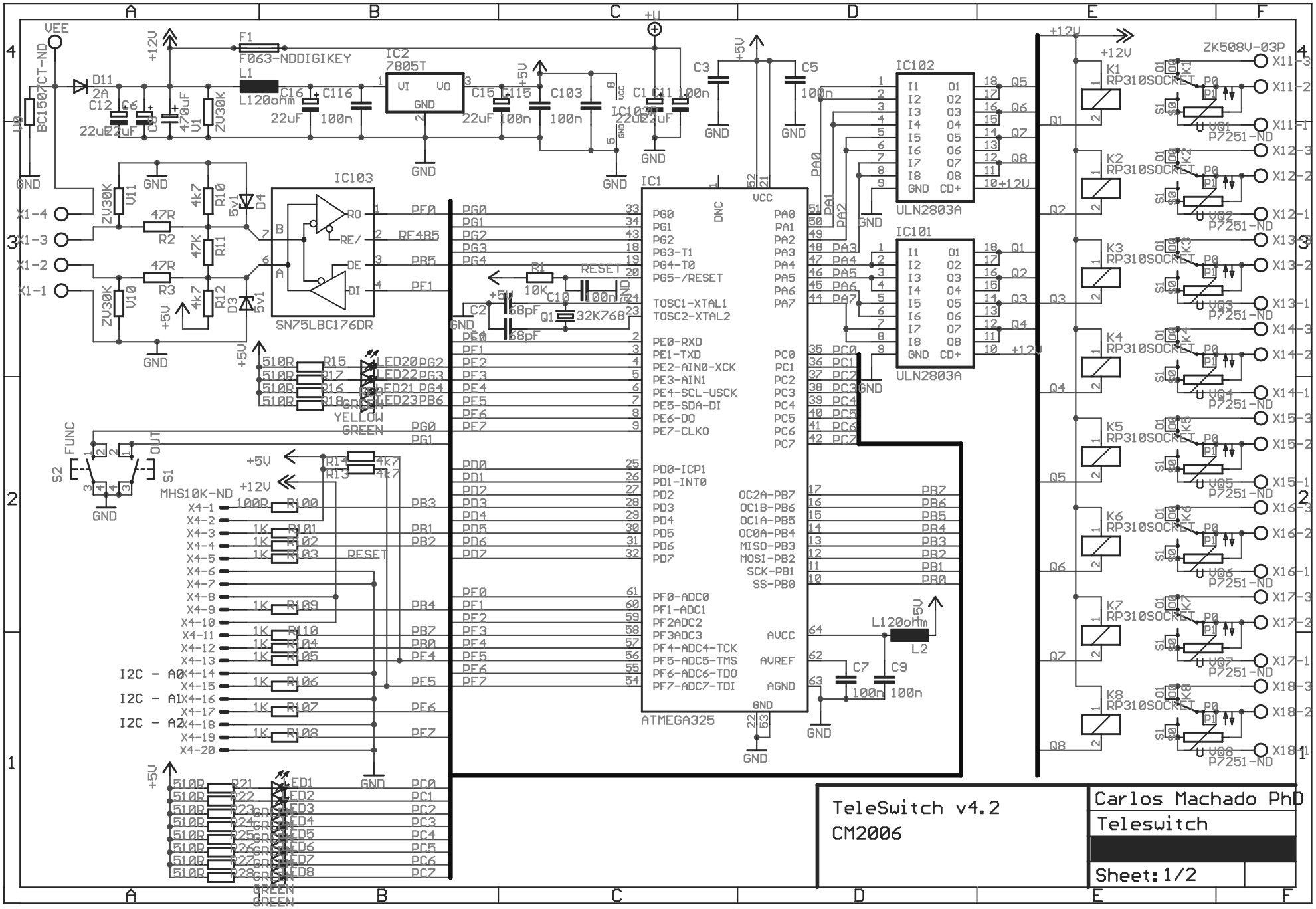
74. [Shalabi et al., 2006] L.A. Shalabi, Z. Shaaban, B. Kasasbeh, ”Data Mining: A Pre-processing Engine”, Journal of Computer Science 2:9 (2006), ISSN: 1549-3636, pp 735-735.

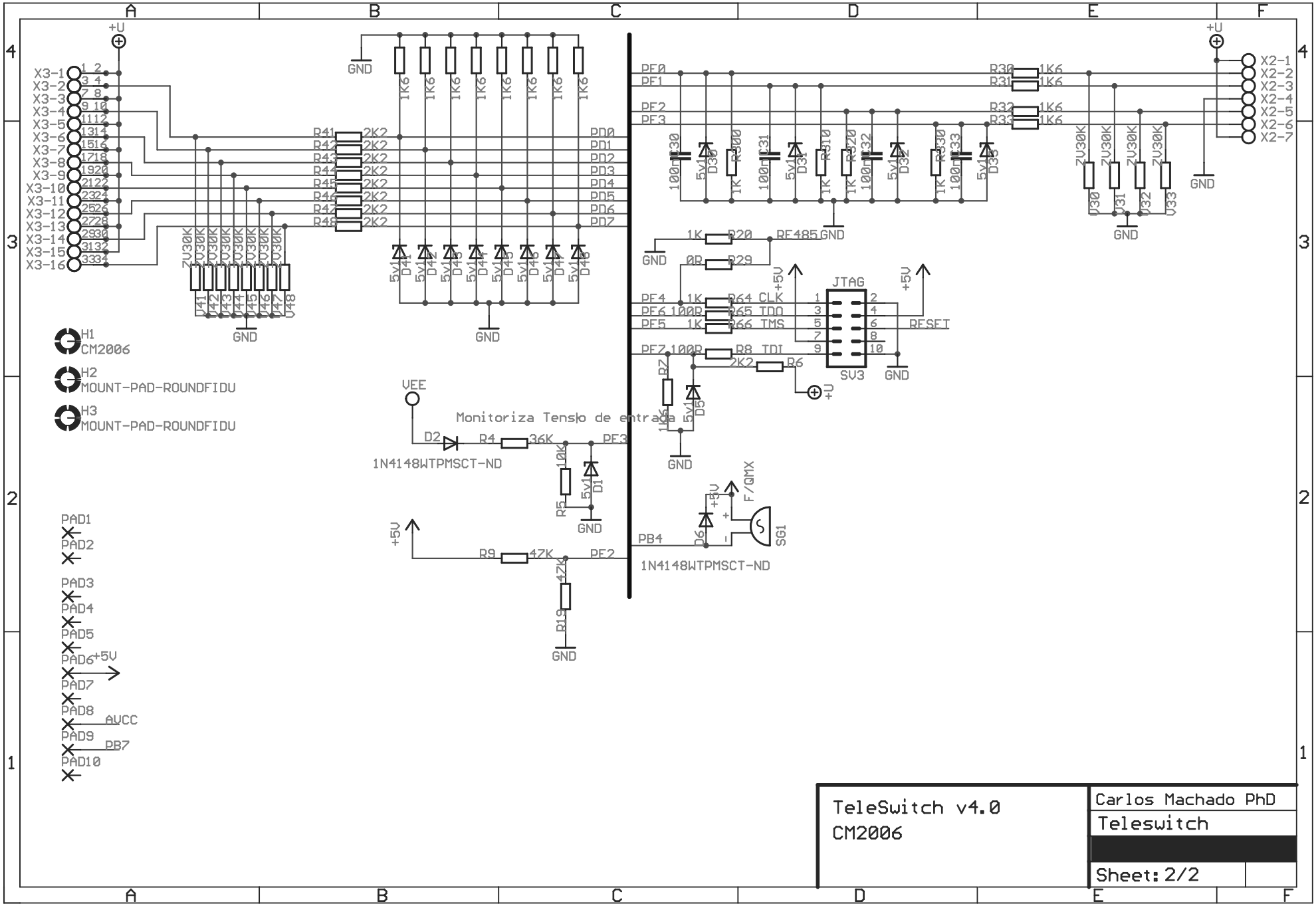
75. [Gurney, 1997] Gurney, K. (1997),”An Introduction to Neural Networks”, London: Routledge. ISBN 1-85728-673-1

76. [Michael et al., 2004] Michael J. A. Berry, Gordon S. Linoff, ”Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management”, 2nd Ed., 2004, ISBN: 978-0-471-47064-9

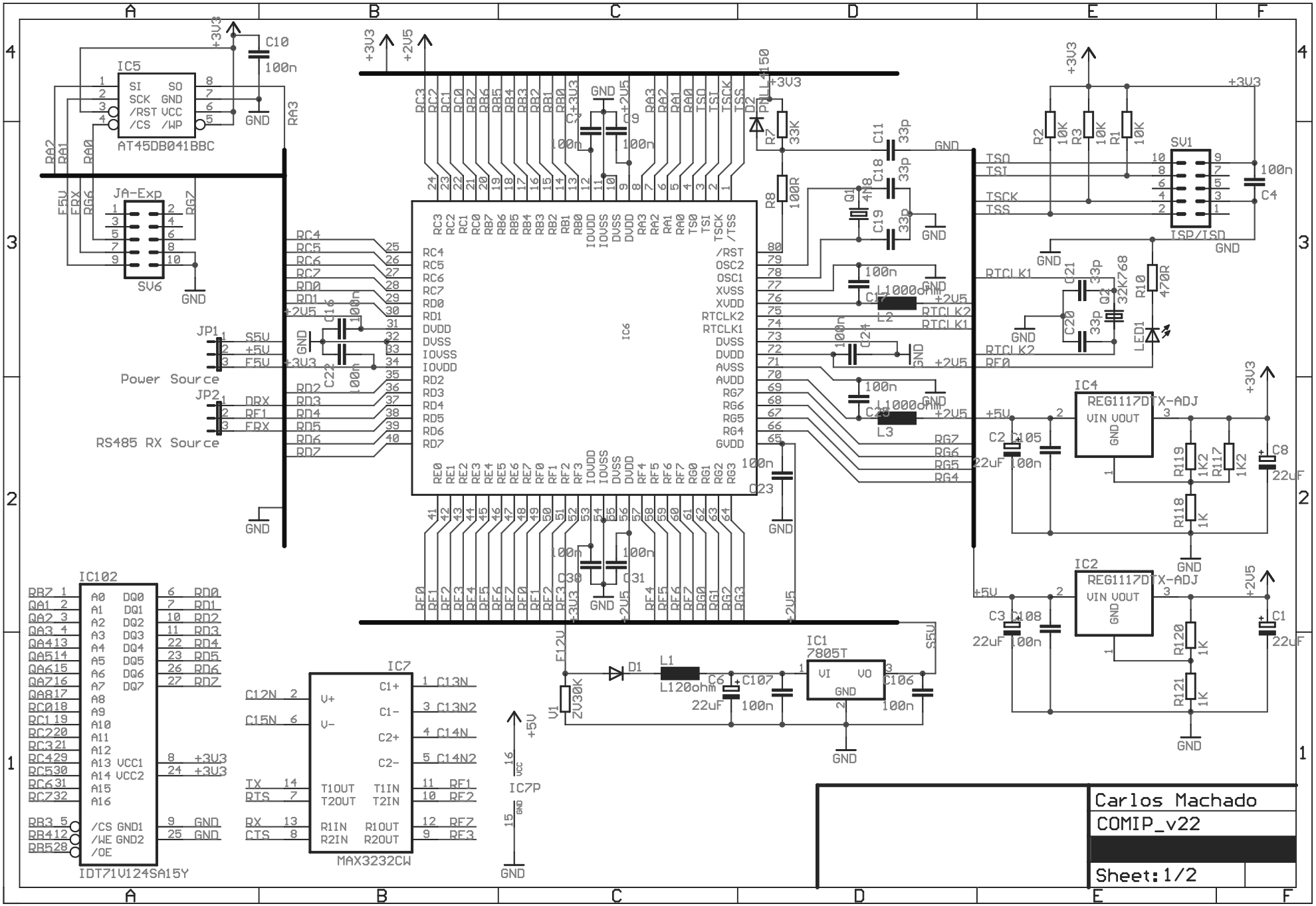
Appendixes

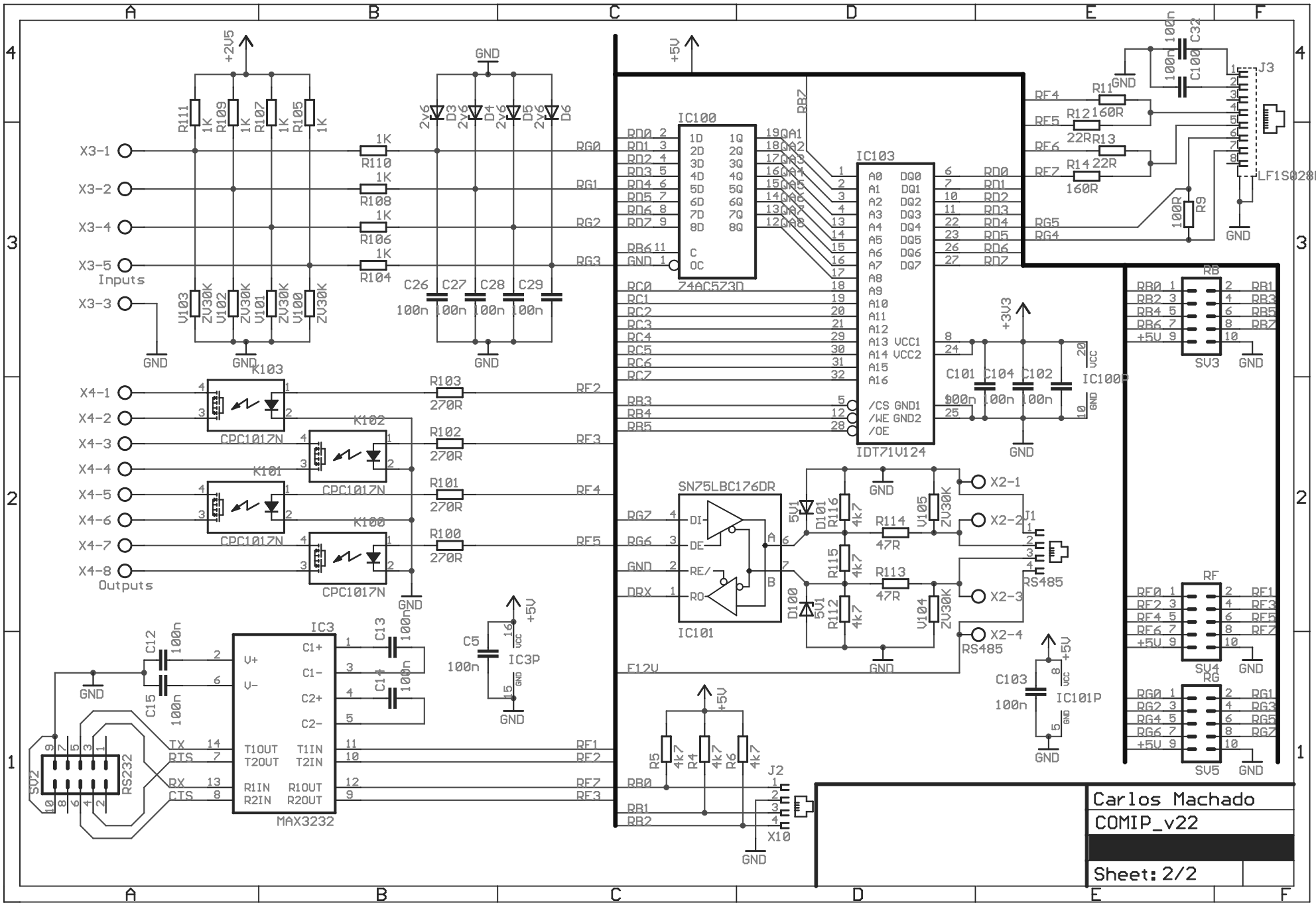
A1. Teleswitch device schematic


TeleSwitch v4.2
CM2006
Carlos Machado PhD
Teleswitch
Sheet: 1/2
ATMEGA325
SN75LBC176DR
ULN2803A
RP310SOCKET
P7251-ND
ZK508V-03P
MHS10K-ND
BC1507CT-ND
32K768
RESET
FUNC
I2C - A0
I2C - A1
I2C - A2


TeleSwitch v4.0
CM2006
Carlos Machado PhD
Teleswitch
Sheet: 2/2
Monitoriza Tensão de entrada
1N4148WTPMSCT-ND
1N4148WTPMSCT-ND
JTAG
SV3
RESET
F/QMX
SG1
H1
CM2006
H2
MOUNT-PAD-ROUNDFIDU
H3
MOUNT-PAD-ROUNDFIDU
PAD1
PAD2
PAD3
PAD4
PAD5
PAD6
PAD7
PAD8
PAD9
PAD10
AVCC
PB7
+5V
GND
VEE

A2. IP Communicator device schematic


IC5
AT45DB041BBC
JA-Exp
SV6
Power Source
JP1
JP2
RS485 RX Source
IC6
IC102
IDT71V124SA15Y
IC7
MAX3232CW
IC7P
IC1
7805T
IC4
REG1117D
IC2
REG1117D
SV1
ISP/ISD
LED1
D1
L1
L120ohm
L1000ohm
Q1
Q2
32K768
ZV30K
Carlos Machado
COMIP_v22
Sheet:1/2


Carlos Machado
COMIP_v22
Sheet: 2/2
IC100
74AC573D
IC103
IDT71V124
IC101
SN75LBC176DR
IC3
MAX3232
IC100P
IC3P
IC101P
RS232
RS485
LF1S028LS
Inputs
Outputs
X3-1
X3-2
X3-4
X3-5
X3-3
X4-1
X4-2
X4-3
X4-4
X4-5
X4-6
X4-7
X4-8
K103
K102
K101
K100
CPC1017N
SV3
SV4
SV5
SV2
J1
J2
J3
X10
X2-1
X2-2
X2-3
X2-4
+5V
+3V3
+2V5
GND
F12V

A3. Atmel329 Datasheet (Microcontroller Features)

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - In-System Self-Programmable Flash, Endurance: 10,000 Write/Erase Cycles
    32K bytes (ATmega329/ATmega3290)
    64K bytes (ATmega649/ATmega6490)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - EEPROM, Endurance: 100,000 Write/Erase Cycles
    1K bytes (ATmega329/ATmega3290)
    2K bytes (ATmega649/ATmega6490)
  - Internal SRAM
    2K bytes (ATmega329/ATmega3290)
    4K bytes (ATmega649/ATmega6490)
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - 4 x 25 Segment LCD Driver (ATmega329/ATmega649)
  - 4 x 40 Segment LCD Driver (ATmega3290/ATmega6490)
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Universal Serial Interface with Start Condition Detector
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 53/68 Programmable I/O Lines
  - 64-lead TQFP, 64-pad QFN/MLF, and 100-lead TQFP
- Speed Grade:
  - ATmega329V/ATmega3290V/ATmega649V/ATmega6490V:
    0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega329/3290/649/6490:
    0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
- Temperature range:
  - -40°C to 85°C Industrial

8-bit AVR® Microcontroller with In-System Programmable Flash

ATmega329/V
ATmega3290/V
ATmega649/V
ATmega6490/V

Preliminary

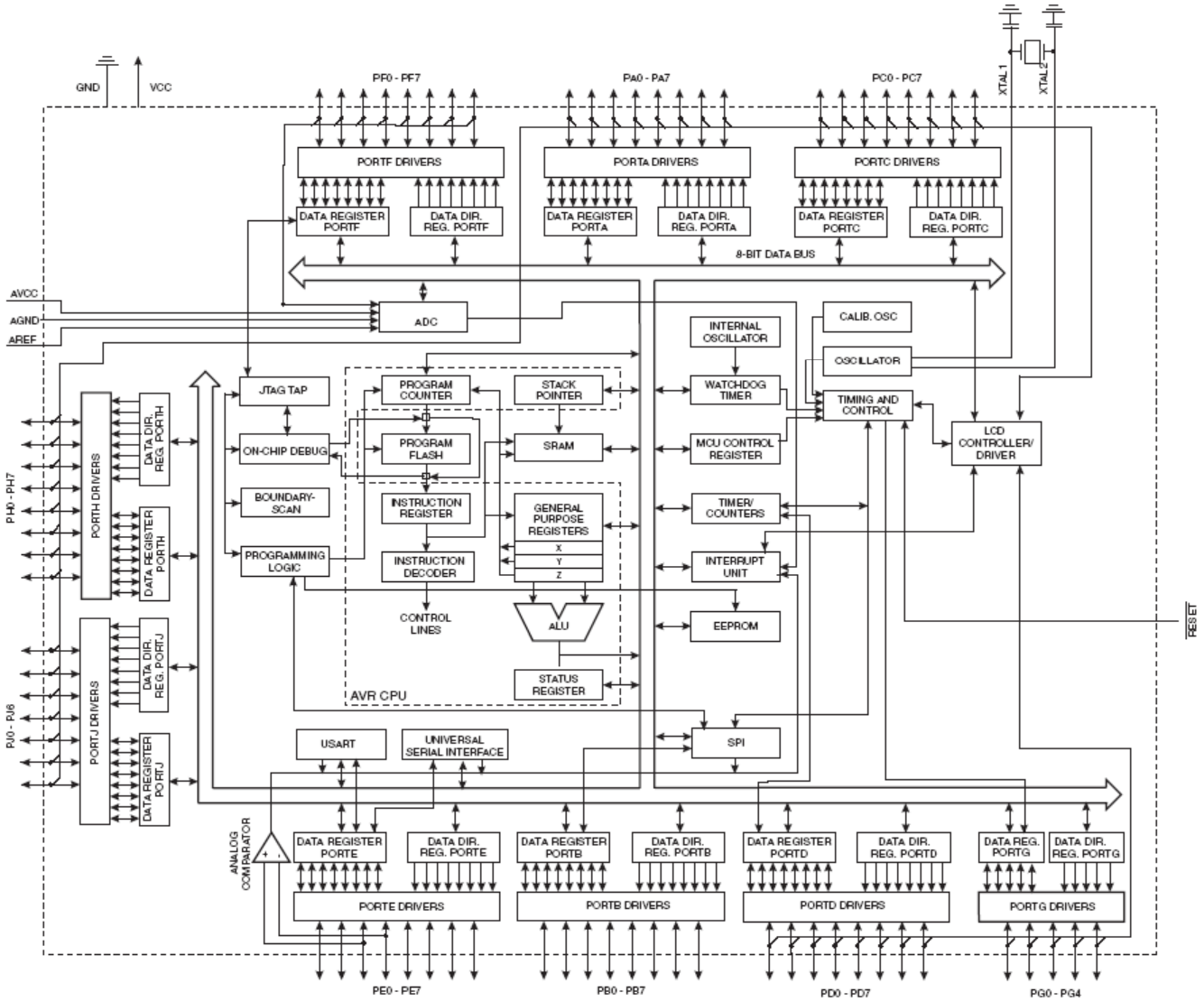
2552C–AVR–03/06

Overview

The ATmega329/3290/649/6490 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega329/3290/649/6490 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 3. Block Diagram



A4. Teleswitch Network (common definition file)

```
…
extern volatile uint8_t net_addr;

// Internal network states values
#define NET_ST_IDLE                0    // Data Bus in IDLE
#define NET_ST_RX_WAITSTART        1    // Wait frame sync
#define NET_ST_RX_BYTES            2    // Receiving data frame
#define NET_ST_RX_COMPLETE         3    // Receive complete (CRC checking)
#define NET_ST_TX_MDACCESS         4    // Data Frame Transmission, waiting medium access
#define NET_ST_TX_TRANSMIT         5    // Transmiting Data Frame
#define NET_ST_TX_VERIFY           6    // Transmission verification

// Internal Network Flags
typedef struct _net_control{
  uint8_t    lock:1;                   // Network medium blocked
  uint8_t    state:3;                  // Network internal state
  uint8_t    byte_count:4;             // Byte Counter
}net_control;

extern volatile net_control net;

#define NET_BROADCAST_ADDR         0xFF
#define NET_GROUP_MASK             0x0F

#define NET_TRAMA_START  0xA5

#define NET_TRAMA_TYPE_MAC         0
#define NET_TRAMA_TYPE_CONTROL     1
#define NET_TRAMA_TYPE_EVENT       2
#define NET_TRAMA_TYPE_BINDING     3
#define NET_TRAMA_TYPE_OUTPUT      4

//For MAC frames – sub type values
#define NET_TRAMA_STYPE_REQUEST          0
#define NET_TRAMA_STYPE_AUTHORIZED       1
#define NET_TRAMA_STYPE_QUERY_ADDR       2
#define NET_TRAMA_STYPE_ADDR_USED        3

// MAC frames data type
typedef struct _net_tr_mac{
  uint8_t    auth_addr;
  uint8_t    data;
  uint16_t   id;
}net_tr_mac;

// CONTROL - frames – sub type values
#define NET_TRAMA_STYPE_ECHOQRY          0
#define NET_TRAMA_STYPE_ECHOACK          1
#define NET_TRAMA_STYPE_VERSION          2
#define NET_TRAMA_STYPE_LOCK             3    // BUS Lock
#define NET_TRAMA_STYPE_BOOT             4    // Reboot and Boot-Loader command
#define NET_TRAMA_STYPE_CHGADDR          5
#define NET_TRAMA_STYPE_STATUS           6
#define NET_TRAMA_STYPE_STATIST_CMD      7
#define NET_TRAMA_STYPE_STATIST_P1       8
#define NET_TRAMA_STYPE_STATIST_P2       9
#define NET_TRAMA_STYPE_STATIST_P3       10


#define BOOT_FUNC_REBOOT           0
#define BOOT_FUNC_BOOTLOAD         1

#define BOOT_RSP_REQUEST           0
#define BOOT_RSP_EXECUTING         1
#define BOOT_RSP_IGNORING          2

#define CHGADDR_RSP_REQUEST        0
#define CHGADDR_RSP_CHANGED        1
#define CHGADDR_RSP_IGNORING       2

#define STATIST_CMD_QUERY          0    // Request Net Statistics
#define STATIST_CMD_SEQQUERY       1    // Sequential Request Net Statistics
#define STATIST_CMD_RESET          2    // Reset Net statistical Counters
#define STATIST_CMD_ALL            0
#define STATIST_CMD_P1             1
```

```
#define STATIST_CMD_P2              2
#define STATIST_CMD_P3              3

// EVENT - frames – sub type values
#define NET_TRAMA_STYPE_LOCAL       0
#define NET_TRAMA_STYPE_REMOTE      1
#define NET_TRAMA_STYPE_OUTPUTACT   2

// BINDING- frames – sub type values
#define NET_TRAMA_STYPE_CMDACK      0   // Request commands and ack
#define NET_TRAMA_STYPE_CMDINSERT   1   // Insert Binding Entry
#define NET_TRAMA_STYPE_CMDREMOVE   2   // Remove Binding Entry
#define NET_TRAMA_STYPE_ENTRY       3   // Binding Entry Data
#define NET_TRAMA_STYPE_USEDSIZE    4   // Binding Table Usage

#define     COMMAND_ACK             0
#define     COMMAND_ACK_INSERT      0
#define     COMMAND_GET_USEDSZ      1
#define     COMMAND_ACK_REMOVE      1
#define     COMMAND_GET_ENTRY       2
#define     COMMAND_LOAD            3
#define     COMMAND_SAVE            4
#define     COMMAND_RESET           5
#define     COMMAND_CLEAN           6

#define     CMD_ACK_RET_OK          0
#define     CMD_ACK_RET_NOTFOUND    1
#define     CMD_ACK_RET_DUPLICATE   2
#define     CMD_ACK_RET_BINDFULL    3
#define     CMD_ACK_RET_BINDADDRFULL 4

// OUTPUT - frames – sub type values
#define NET_TRAMA_STYPE_GET_STATE   0

#define GET_STATE_ACK               0
#define GET_STATE_RELAYS            1

#define GET_STATE_BANK0             0


//---------- Network Type Structures
typedef struct          _net_tr_control{
  uint8_t   data[4];
}net_tr_control;

typedef struct          _net_tr_control16{
  uint16_t  data[2];
}net_tr_control16;

typedef struct          _net_tr_event{
  uint8_t   io_id;                  // IO Identifications
  uint8_t   io_type:4;              // Type of IO
  uint8_t   bind_data:4;            // Binding Data Information (fixed data)
  uint8_t   data;                   // IO extra information
}net_tr_event;

typedef struct          _net_tr_binding_cmdack{
  uint8_t               cmd;        // Command
  uint8_t               param[3];   // Command Parameters
}net_tr_binding_cmdack;

typedef struct          _net_tr_binding_entry{
  uint8_t   addr;                   // Device address for the binding
  uint8_t   output;                 // Output Identification
  uint8_t   action:4;               // Action Type
  uint8_t   bind_data:4;            // Extra binding data
  uint8_t   event;                  // Event Identification (tipicaly IO identification)
}net_tr_binding_entry;

typedef struct          _net_tr_outputs_get_state{
  uint8_t               cmd:4;      // Command
  uint8_t               bank:4;     // Bank to read
  uint8_t               outstat;    // Relay states
  uint8_t               param[2];   // extra info
}net_tr_outputs_get_state;
```

```
typedef struct _net_trama{
  uint8_t    start;                              // Sync byte for the frame
  uint8_t    dst_addr;                           // Destination address
  uint8_t    src_addr;                           // Source address
  uint8_t    tipo:4;                             // Frame type
  uint8_t    subtipo:4;                          // Frame sub-type
  union {
             net_tr_mac                 mac;
             net_tr_control             control;
             net_tr_control16           control16;
             net_tr_event               event;
             net_tr_binding_cmdack      bind_cmd;
             net_tr_binding_entry       bind_entry;
             net_tr_outputs_get_state   outputs_stat;
  };
  uint16_t   CRC;                                // Frame CRC (16 bit)
}net_trama;

typedef struct _net_trama_data{
  union{
             uint8_t    rawdata[sizeof(net_trama)];
             net_tramadata;
  };
}net_trama_data;
…
typedef struct _net_statistics{
  uint16_t   rx_total;                  // Total frames received successfully
  uint16_t   rx_sync_throwout;          // Total bytes ignored before frame sync
  uint16_t   rx_processed;              // Total frames processed
  uint16_t   tx_total;                  // Total frames sent

  uint8_t    rx_crc_error;              // Total frames received with CRC error
  uint8_t    rx_timeout;                // Total receiving timeouts
  uint8_t    rx_throwout;               // Total frames throw out because of low resources
  uint8_t    tx_colision;               // Total Transmission collisions

}net_statistics;


//-------------------------------------------------------
// (MAC) Medium reservation for Broadcast without collisions
//
#define MAC_REQUEST_TIME                4       // 20 ms *4
#define MAC_RESERVED_TIME               4       // reserved time multiplier
#define MAC_WAIT_REPLY                  24      // wait time until new address usage

#define MAC_ST_IDLE                     0       // Reservation IDLE.  MAC in CSMA mode
#define MAC_ST_WAIT_SEND_REQUEST        1       // Request medium reservation
#define MAC_ST_WAIT_ACCESS              2       // Waiting medium  reservation
#define MAC_ST_WAIT_SEND_AUTHOR         3       //
#define MAC_ST_MEDIUM_AUTHORIZED        4       // Bus reserved. Broadcast can start
#define MAC_ST_MEDIUM_RESERVED          5       // Bus reserved by other node

#define MAC_ST_WAIT_SEND_QUERY          6       // Query address usage
#define MAC_ST_WAIT_REPLY               7       // Waiting address usage reply


typedef struct _MAC_control{
  uint8_t    state:3;                   // MAC control state
  uint8_t    notused:2;
  uint8_t    addr_conf:1;               // Address conflit detected
  uint8_t    tramasend:1;               // MAC control frame sent
  uint8_t    coord:1;                   // The device has the bus coordination
}MAC_Control;

extern volatile MAC_Control mac_ctrl;
```

A5. UDOMUS XML Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<DOMUS>
  <SYSTEMS>
		<TELESWITCH>
			<GATEWAY>
				<URL><![CDATA[http://192.168.0.99/]]></URL>
			</GATEWAY>
		</TELESWITCH>
  </SYSTEMS>
  <SPACE ID="APARTAMENTO">
		<TITLE>APARTAMENTO</TITLE>
		<IMAGE>img_home/apartamento.jpg</IMAGE>
		<ACTION>
			<TITLE>APAGAR TUDO</TITLE>
			<SYSTEM>TELESWITCH</SYSTEM>
			<COMMAND>
				<TYPE>OFF</TYPE>
				<EXECUTE>
					<![CDATA[swt?c=A&p=12001&a=12]]>
				</EXECUTE>
			</COMMAND>
		</ACTION>
		<ACTION>
			<TITLE>ILUMINAÇÃO DE CHEGADA</TITLE>
			<SYSTEM>TELESWITCH</SYSTEM>
			<COMMAND>
				<TYPE>ON</TYPE>
				<EXECUTE>
					<![CDATA[swt?c=A&p=12001&a=10]]>
				</EXECUTE>
			</COMMAND>
		</ACTION>
		<POSITION><![CDATA[left:41px; top:242px;]]></POSITION>
		<COORDS>41,242, 89,242, 89,290, 41,290,41,242</COORDS>
  </SPACE>
  <SPACE ID="SUITE">
		<TITLE>SUITE</TITLE>
		<IMAGE>img_home/suite.JPG</IMAGE>
		<ACTION>
			<TITLE>ILUMINAÇÃO DA ENTRADA</TITLE>
			<SYSTEM>TELESWITCH</SYSTEM>
			<FEEDBACK><ADDR>1000</ADDR></FEEDBACK>
			<COMMAND>
				<TYPE>ON</TYPE>
				<EXECUTE>
					<![CDATA[swt?c=A&p=10802&a=2]]>
				</EXECUTE>
				<FEEDBACK IOMASK="0x01" STMASK="1"></FEEDBACK>
			</COMMAND>
			<COMMAND>
				<TYPE>OFF</TYPE>
				<EXECUTE>
					<![CDATA[swt?c=A&p=10802&a=1]]>
				</EXECUTE>
				<FEEDBACK IOMASK="0x01" STMASK="0"></FEEDBACK>
			</COMMAND>
		</ACTION>
		<ACTION>
			<TITLE>ILUMINAÇÃO PRINCIPAL</TITLE>
			<SYSTEM>TELESWITCH</SYSTEM>
			<FEEDBACK><ADDR>1000</ADDR></FEEDBACK>
			<COMMAND>
				<TYPE>ON</TYPE>
				<EXECUTE>
					<![CDATA[swt?c=A&p=10812&a=2]]>
				</EXECUTE>
				<FEEDBACK IOMASK="0x02" STMASK="0x02"></FEEDBACK>
			</COMMAND>
			<COMMAND>
				<TYPE>OFF</TYPE>
				<EXECUTE>
					<![CDATA[swt?c=A&p=10812&a=1]]>
				</EXECUTE>
				<FEEDBACK IOMASK="0x02" STMASK="0x00"></FEEDBACK>
```

```
                </COMMAND>
        </ACTION>
        <POSITION><![CDATA[left:169px; top:65px;]]></POSITION>
        <COORDS>166,184, 223,184, 223,65, 323,65, 323,217, 169,217, 166,184</COORDS>
</SPACE>
<SPACE ID="CORREDOR">
        <TITLE>CORREDOR</TITLE>
        <IMAGE>img_home/corredor.JPG</IMAGE>
        <ACTION>
                <TITLE>ILUMINAÇÃO</TITLE>
                <SYSTEM>TELESWITCH</SYSTEM>
                <FEEDBACK><ADDR>1200</ADDR></FEEDBACK>
                <COMMAND>
                        <TYPE>ON</TYPE>
                        <EXECUTE>
                                <![CDATA[swt?c=A&p=12812&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x02" STMASK="0x02"></FEEDBACK>
                </COMMAND>
                <COMMAND>
                        <TYPE>HALF</TYPE>
                        <EXECUTE>
                                <![CDATA[swt?c=A&p=12812&a=1&c=A&p=12822&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x06" STMASK="0x04"></FEEDBACK>
                </COMMAND>
                <COMMAND>
                        <TYPE>OFF</TYPE>
                        <EXECUTE>
                                <![CDATA[swt?c=A&p=12812&a=1&c=A&p=12822&a=1]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x06" STMASK="0x00"></FEEDBACK>
                </COMMAND>
        </ACTION>
        <POSITION><![CDATA[left:116px; top:162px;]]></POSITION>
        <COORDS>121,162, 164,162, 164,323, 122,320, 121,195, 115,192, 115,164, 121,162</COORDS>
</SPACE>
<SPACE ID="ESCRITORIO">
        <TITLE>ESCRITÓRIO</TITLE>
        <IMAGE>img_home/escritorio.JPG</IMAGE>
        <ACTION>
                <TITLE>ILUMINAÇÃO</TITLE>
                <SYSTEM>TELESWITCH</SYSTEM>
                <FEEDBACK><ADDR>1000</ADDR></FEEDBACK>
                <COMMAND>
                        <TYPE>ON</TYPE>
                        <EXECUTE>
                                <![CDATA[swt?c=A&p=10852&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x20" STMASK="0x20"></FEEDBACK>
                </COMMAND>
                <COMMAND>
                        <TYPE>OFF</TYPE>
                        <EXECUTE>
                                <![CDATA[swt?c=A&p=10852&a=1]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x20" STMASK="0"></FEEDBACK>
                </COMMAND>
        </ACTION>
        <POSITION><![CDATA[left:16px; top:19px;]]></POSITION>
        <COORDS>16,20, 110,20, 110,165, 113,165, 113,194, 16,194, 16,20</COORDS>
</SPACE>
<SPACE ID="QUARTO">
        <TITLE>QUARTO</TITLE>
        <IMAGE>img_home/quarto.JPG</IMAGE>
        <ACTION>
                <TITLE>ILUMINAÇÃO</TITLE>
                <SYSTEM>TELESWITCH</SYSTEM>
                <FEEDBACK><ADDR>1000</ADDR></FEEDBACK>
                <COMMAND>
                        <TYPE>ON</TYPE>
                        <EXECUTE>
                                <![CDATA[swt?c=A&p=10872&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x80" STMASK="0x80"></FEEDBACK>
                </COMMAND>
```

```
                        <COMMAND>
                                <TYPE>HALF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=10872&a=1&c=A&p=10862&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0xC0" STMASK="0x40"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=10862&a=1&c=A&p=10872&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0xC0" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <POSITION><![CDATA[left:120px; top:19px;]]></POSITION>
                <COORDS>120,20,  226,20,  226,59,  212,59,  212,157,  171,157,  171,155,  147,155,  147,158,
120,158, 120,156, 119,155, 119,20, 120,20</COORDS>
        </SPACE>
        <SPACE ID="HALL">
                <TITLE>HALL</TITLE>
                <IMAGE>img_home/hall.JPG</IMAGE>
                <ACTION>
                        <TITLE>ILUMINAÇÃO</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
                        <FEEDBACK><ADDR>1200</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12852&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x20" STMASK="0x20"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>HALF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12852&a=1&c=A&p=12842&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x30" STMASK="0x10"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12852&a=1&c=A&p=12842&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x30" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <POSITION><![CDATA[left:123px; top:325px;]]></POSITION>
                <COORDS>123,326, 153,326, 153,329, 204,329, 204,395, 123,395, 123,326</COORDS>
        </SPACE>
        <SPACE ID="VARANDNORTE">
                <TITLE>VARANDA NORTE</TITLE>
                <IMAGE>img_home/varandnorte.JPG</IMAGE>
                <ACTION>
                        <TITLE>ILUMINAÇÃO</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
                        <FEEDBACK><ADDR>1000</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=10842&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x10" STMASK="0x10"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=10842&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x10" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <POSITION><![CDATA[left:237px; top:18px;]]></POSITION>
                <COORDS>237,18, 323,18, 323,62, 237,62, 237,18, 237,18</COORDS>
        </SPACE>
```

```
<SPACE ID="VARANDSUL">
	<TITLE>VARANDA SUL</TITLE>
	<IMAGE>img_home/varandsul.JPG</IMAGE>
	<ACTION>
		<TITLE>ILUMINAÇÃO</TITLE>
		<SYSTEM>TELESWITCH</SYSTEM>
		<FEEDBACK><ADDR>1100</ADDR></FEEDBACK>
		<COMMAND>
			<TYPE>ON</TYPE>
			<EXECUTE>
				<![CDATA[swt?c=A&p=11822&a=2]]>
			</EXECUTE>
			<FEEDBACK IOMASK="0x04" STMASK="0x04"></FEEDBACK>
		</COMMAND>
		<COMMAND>
			<TYPE>OFF</TYPE>
			<EXECUTE>
				<![CDATA[swt?c=A&p=11822&a=1]]>
			</EXECUTE>
			<FEEDBACK IOMASK="0x04" STMASK="0x00"></FEEDBACK>
		</COMMAND>
	</ACTION>
	<POSITION><![CDATA[left:241px; top:576px;]]></POSITION>
	<COORDS>241,576, 324,576, 324,615, 241,615, 241,576, 241,576</COORDS>
</SPACE>
<SPACE ID="ENTRADA">
	<TITLE>ENTRADA</TITLE>
	<IMAGE>img_home/entrada.JPG</IMAGE>
	<ACTION>
		<TITLE>ILUMINAÇÃO</TITLE>
		<SYSTEM>TELESWITCH</SYSTEM>
		<FEEDBACK><ADDR>1200</ADDR></FEEDBACK>
		<COMMAND>
			<TYPE>ON</TYPE>
			<EXECUTE>
				<![CDATA[swt?c=A&p=12802&a=2]]>
			</EXECUTE>
			<FEEDBACK IOMASK="0x01" STMASK="0x01"></FEEDBACK>
		</COMMAND>
		<COMMAND>
			<TYPE>OFF</TYPE>
			<EXECUTE>
				<![CDATA[swt?c=A&p=12802&a=1]]>
			</EXECUTE>
			<FEEDBACK IOMASK="0x01" STMASK="0x00"></FEEDBACK>
		</COMMAND>
	</ACTION>
	<POSITION><![CDATA[left:15px; top:337px;]]></POSITION>
	<COORDS>15,335, 113,335, 113,381, 54,381, 54,397, 15,397, 15,335, 15,335</COORDS>
</SPACE>
<SPACE ID="WCSUITE">
	<TITLE>WC SUITE</TITLE>
	<IMAGE>img_home/wcsuite.JPG</IMAGE>
	<ACTION>
		<TITLE>ILUMINAÇÃO DO ESPELHO</TITLE>
		<SYSTEM>TELESWITCH</SYSTEM>
		<FEEDBACK><ADDR>1000</ADDR></FEEDBACK>
		<COMMAND>
			<TYPE>ON</TYPE>
			<EXECUTE>
				<![CDATA[swt?c=A&p=10822&a=2]]>
			</EXECUTE>
			<FEEDBACK IOMASK="0x04" STMASK="0x04"></FEEDBACK>
		</COMMAND>
		<COMMAND>
			<TYPE>OFF</TYPE>
			<EXECUTE>
				<![CDATA[swt?c=A&p=10822&a=1]]>
			</EXECUTE>
			<FEEDBACK IOMASK="0x04" STMASK="0x00"></FEEDBACK>
		</COMMAND>
	</ACTION>
	<ACTION>
		<TITLE>ILUMINAÇÃO PRINCIPAL</TITLE>
		<SYSTEM>TELESWITCH</SYSTEM>
		<FEEDBACK><ADDR>1000</ADDR></FEEDBACK>
```

```
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=10832&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x08" STMASK="0x08"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=10832&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x08" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <POSITION><![CDATA[left:267px; top:222px;]]></POSITION>
                <COORDS>267,298, 267,224, 324,224, 324,298, 267,298, 267,298</COORDS>
        </SPACE>
        <SPACE ID="WC">
                <TITLE>WC</TITLE>
                <IMAGE>img_home/wc.JPG</IMAGE>
                <ACTION>
                        <TITLE>ILUMINAÇÃO DO ESPELHO</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
                        <FEEDBACK><ADDR>1200</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12862&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x40" STMASK="0x40"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>HALF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12862&a=1&c=A&p=12872&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0xc0" STMASK="0x80"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12862&a=1&c=A&p=12872&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0xc0" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <ACTION>
                        <TITLE>ILUMINAÇÃO PRINCIPAL</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
                        <FEEDBACK><ADDR>1200</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12832&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x08" STMASK="0x08"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=12832&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x08" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <POSITION><![CDATA[left:170px; top:248px;]]></POSITION>
                <COORDS>173,248, 258,248, 258,319, 171,319, 171,294, 169,294, 169,269, 172,269, 173,248,
173,248</COORDS>
        </SPACE>
        <SPACE ID="COZINHA">
                <TITLE>COZINHA</TITLE>
                <IMAGE>img_home/cozinha.JPG</IMAGE>
                <ACTION>
                        <TITLE>ILUMINAÇÃO PRINCIPAL</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
```

```
                        <FEEDBACK><ADDR>1100</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11872&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x80" STMASK="0x80"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11872&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x80" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <ACTION>
                        <TITLE>ILUMINAÇÃO DA BANCA</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
                        <FEEDBACK><ADDR>1100</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11862&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x40" STMASK="0x40"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11862&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x40" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <ACTION>
                        <TITLE>ILUMINAÇÃO DA LAVANDARIA</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
                        <FEEDBACK><ADDR>1100</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11852&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x20" STMASK="0x20"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11852&a=1]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x20" STMASK="0x00"></FEEDBACK>
                        </COMMAND>
                </ACTION>
                <POSITION><![CDATA[left:124px; top:399px;]]></POSITION>
                <COORDS>145,402, 174,402, 174,398, 200,398, 202,402, 202,621, 124,621, 124,414, 145,413,
145,402</COORDS>
        </SPACE>
        <SPACE ID="SALA">
                <TITLE>SALA</TITLE>
                <IMAGE>img_home/sala.JPG</IMAGE>
                <ACTION>
                        <TITLE>ILUMINAÇÃO SALA DE ESTAR</TITLE>
                        <SYSTEM>TELESWITCH</SYSTEM>
                        <FEEDBACK><ADDR>1100</ADDR></FEEDBACK>
                        <COMMAND>
                                <TYPE>ON</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11842&a=2]]>
                                </EXECUTE>
                                <FEEDBACK IOMASK="0x10" STMASK="0x10"></FEEDBACK>
                        </COMMAND>
                        <COMMAND>
                                <TYPE>OFF</TYPE>
                                <EXECUTE>
                                        <![CDATA[swt?c=A&p=11842&a=1]]>
```

```
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x10" STMASK="0x00"></FEEDBACK>
                    </COMMAND>
                </ACTION>
                <ACTION>
                    <TITLE>ILUMINAÇÃO INDIRETA</TITLE>
                    <SYSTEM>TELESWITCH</SYSTEM>
                    <FEEDBACK><ADDR>1100</ADDR></FEEDBACK>
                    <COMMAND>
                        <TYPE>ON</TYPE>
                        <EXECUTE>
                            <![CDATA[swt?c=A&p=11832&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x08" STMASK="0x08"></FEEDBACK>
                    </COMMAND>
                    <COMMAND>
                        <TYPE>OFF</TYPE>
                        <EXECUTE>
                            <![CDATA[swt?c=A&p=11832&a=1]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x08" STMASK="0x00"></FEEDBACK>
                    </COMMAND>
                </ACTION>
                <ACTION>
                    <TITLE>ILUMINAÇÃO AMBIENTE</TITLE>
                    <SYSTEM>TELESWITCH</SYSTEM>
                    <FEEDBACK><ADDR>1300</ADDR></FEEDBACK>
                    <COMMAND>
                        <TYPE>ON</TYPE>
                        <EXECUTE>
                            <![CDATA[swt?c=A&p=13802&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x01" STMASK="0x01"></FEEDBACK>
                    </COMMAND>
                    <COMMAND>
                        <TYPE>OFF</TYPE>
                        <EXECUTE>
                            <![CDATA[swt?c=A&p=13802&a=1]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x01" STMASK="0x00"></FEEDBACK>
                    </COMMAND>
                </ACTION>
                <ACTION>
                    <TITLE>ILUMINAÇÃO MESA DE JANTAR</TITLE>
                    <SYSTEM>TELESWITCH</SYSTEM>
                    <FEEDBACK><ADDR>1100</ADDR></FEEDBACK>
                    <COMMAND>
                        <TYPE>ON</TYPE>
                        <EXECUTE>
                            <![CDATA[swt?c=A&p=11802&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x01" STMASK="0x01"></FEEDBACK>
                    </COMMAND>
                    <COMMAND>
                        <TYPE>HALF</TYPE>
                        <EXECUTE>
                            <![CDATA[swt?c=A&p=11802&a=1&c=A&p=11812&a=2]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x03" STMASK="0x02"></FEEDBACK>
                    </COMMAND>
                    <COMMAND>
                        <TYPE>OFF</TYPE>
                        <EXECUTE>
                            <![CDATA[swt?c=A&p=11802&a=1&c=A&p=11812&a=1]]>
                        </EXECUTE>
                        <FEEDBACK IOMASK="0x03" STMASK="0x00"></FEEDBACK>
                    </COMMAND>
                </ACTION>
                <POSITION><![CDATA[left:208px; top:307px;]]></POSITION>
                <COORDS>209,328, 280,328, 280,307, 304,306, 304,328, 325,328, 324,571, 209,571, 209,386,
209,327, 209,328</COORDS>
            </SPACE>
            <COMMANDS>
                <TYPE>
                    <ON>
                        <IMAGE>img_sup/bt_on.gif</IMAGE>
```

```
                        <AIMAGE>img_sup/bt_on_r.gif</AIMAGE>
                        <OIMAGE>img_sup/bt_on_g.gif</OIMAGE>
                        <POWER>100</POWER>
                </ON>
                <HALF>
                        <IMAGE>img_sup/bt_half.gif</IMAGE>
                        <AIMAGE>img_sup/bt_half_r.gif</AIMAGE>
                        <OIMAGE>img_sup/bt_half_g.gif</OIMAGE>
                        <POWER>50</POWER>
                </HALF>
                <OFF>
                        <IMAGE>img_sup/bt_off.gif</IMAGE>
                        <AIMAGE>img_sup/bt_off_r.gif</AIMAGE>
                        <OIMAGE>img_sup/bt_off_g.gif</OIMAGE>
                        <POWER>0</POWER>
                </OFF>
        </TYPE>
  </COMMANDS>
</DOMUS>
```